Markt&Technik

Thomas Erpel

CPC BASIC-Kurs

Ein BASIC-Lehrgang für Erstanwender:
- Strukturiertes Programmieren
- Befehlsübersicht
- Unterhaltsame Beispiele

CPC BASIC-Kurs

Thomas Erpel

CPC BASIC-Kurs

Ein BASIC-Lehrgang für Erstanwender:

- Strukturiertes Programmieren
- Befehlsübersicht
- Unterhaltsame Beispiele

Markt & Technik Verlag

CIP-Kurztitelaufnahme der Deutschen Bibliothek

Erpel, Thomas:
CPC-BASIC-Kurs : e. BASIC-Lehrgang für Erstanwender:
strukturiertes Programmieren, Befehlsübersicht, unterhaltsame Beispiele / Thomas Erpel. —
Haar bei München : Markt-und-Technik-Verlag, 1985.
ISBN 3-89090-167-0

15 14 13 12 11 10 9 8 7 6 5 4 3 2 1
89 88 87 86

ISBN 3-89090-167-0

Einbandgestaltung: Grafikdesign Heinz Rauner
Druck: Jantsch, Günzburg
Printed in Germany

Inhaltsverzeichnis

1 Einleitung

Homecomputer erfreuen sich einer immer größeren Beliebtheit. Ihre Leistung und der oft erstaunlich günstige Preis machen sie für viele Anwender interessant.

Als ich Anfang 1984 in einer englischen Zeitschrift von einem Computersystem mit Supereigenschaften und einem durchaus erschwinglichen Preis las, war ich gespannt, wann dieses System auch in Deutschland erhältlich sein würde. Im Spätsommer war es dann so weit, und ich erwarb unter größten Schwierigkeiten eines der ersten verfügbaren Modelle.

Da ich mich intensiv mit den Grafikmöglichkeiten beschäftigen wollte, wählte ich das Modell mit dem Farbmonitor.

Für einen erstaunlich günstigen Preis bekam ich ein komplettes Kleincomputersystem mit Farbmonitor, dessen technische Eigenschaften so manchen Computerfreund aufhorchen ließen:

- leistungsfähiges BASIC
- 20-, 40- oder 80-Zeichen-Darstellung
- 27 verschiedene Farben
- ausgezeichnete Grafikfähigkeiten durch hohe Auflösung
- Echtzeitfunktionen
- Window-Technik

Dazu ist der Schneider CPC 464 auch noch CP/M-fähig, was ein umfangreiches Software-Angebot mit einem Schlag verfügbar macht. Außerdem ist die Speicherkapazität nach oben ausbaufähig, was für Erweiterungen sehr wichtig ist.

Mit einem Diskettenlaufwerk augestattet, ist auch ein kommerzieller Einsatz des CPC 464 möglich. Gerade für Aufgaben wie Rechnungsschreibung, Datenspeicherung und Textverarbeitung bietet sich dieses System für Selbständige und Kleinbetriebe an.

Um alle Leistungen eines solchen Systems voll ausschöpfen zu können, benötigt der Einsteiger einen Ratgeber, der ihn Schritt für Schritt in die neue Materie "Computer" einarbeitet.

Das vorliegende Anwenderhandbuch für den Schneider CPC 464 ist als Einstiegshilfe für Anfänger gedacht, die das Buch auch später noch als Arbeitshilfe benutzen möchten. Behandelt werden darin sowohl die Bedienung als auch die Programmierung des Systems, wobei keinerlei Vorkenntnisse nötig sind.

In den ersten Kapiteln erlernen Sie die Grundregeln für Bedienung und Programmierung. Weiter geht es dann mit Programmiertechniken, Tonerzeugung und Grafik.

Die letzten Kapitel behandeln schließlich den Einsatz in der Praxis und sind vor allem für diejenigen interessant, die sich bereits ein wenig mit der Programmiersprache BASIC auskennen und selbst schon Programme schreiben können. Hier gibt es viele Tips und Tricks zum Schneider, ein Kapitel mit fertig programmierten Programmbausteinen und einiges mehr.

Alles zusammen gesehen ist dies also ein Buch, das neben dem Originalhandbuch von Schneider mit auf den Arbeitstisch gehört.

Sollten Sie irgendwelche Verbesserungsvorschläge haben, wäre ich über eine entsprechende Mitteilung dankbar.

Ich wünsche Ihnen mit diesem Handbuch viel Spaß und Erfolg bei der Arbeit mit Ihrem Schneider CPC 464.

Gütersloh, im Januar 1985 Thomas Erpel

2 Der erste Kontakt

Wenn Sie sich den CPC 464 zugelegt haben, möchten Sie sich natürlich auch sofort an die Arbeit machen.

Packen Sie also das Gerät aus und stellen Sie es vor sich auf den Tisch. In der Styroporverpackung des Tastaturgehäuses befindet sich eine Demokassette, die Sie mit der A-Seite nach oben in den Datenrekorder legen können.

Vielleicht suchen Sie vergeblich nach einem Netzgerät für den Computer, wie man es beispielsweise von anderen Systemen her kennt. Keine Angst, das Netzteil wurde nicht etwa vergessen! Es befindet sich fest eingebaut in dem mitgelieferten Datenmonitor. Sie müssen also nur noch das Monitorkabel und die Spannungsversorgung in Ihren Schneider stecken.

Vertauschen können Sie die beiden Anschlüsse nicht, denn bei dem Monitorkabel handelt es sich um eine fünfpolige Steckverbindung und bei der Stromversorgung um einen sogenannten Klinkenstecker.

Schalten Sie nun am Monitor das gesamte System ein.

Am Tastaturgehäuse befindet sich rechts noch ein kleiner Schiebeschalter, den Sie in die richtige Position bringen müssen, damit auch der Rechner selbst mit Spannung versorgt wird. Allerdings stellt dieser Schalter nach meiner Ansicht ein Nachteil dar, denn Schiebeschalter neigen bei längerem Gebrauch eher zu Kontaktschäden als beispielsweise Kippschalter.

Wenn alles richtig angeschlossen ist, sehen Sie die Systemmeldung auf dem Monitor. Diese besagt lediglich, von welchem Hersteller die Hardware und von wem die eingebaute Software stammt. Außerdem wird meistens angegeben, welche Speichergröße das System hat. Viel wichtiger als die Systemmeldung ist das Wort READY und ein kleines, helles Quadrat unter dem Wort READY. Dies bedeutet für Sie als Benutzer, daß der Schneider nun bereit ist und auf Ihre Eingaben wartet.

Das Quadrat hat einen speziellen Namen; man nennt es *Cursor*. Der Cursor gibt die *aktuelle Schreibposition* auf dem Bildschirm an und zeigt Ihnen später, zusammen mit dem READY, daß der Rechner seine Aufgaben

einwandfrei abgearbeitet hat. Sie werden sich recht schnell an diese Darstellung gewöhnen und bald bemerken, wenn irgendetwas nicht stimmt.

Im folgenden geht es nun um die Tastaturbelegung des CPC 464, die ich Ihnen anhand einiger Beispiele erläutern werde. an einigen Beispielen erklären.

3 Die Tastatur des Schneiders

Ihr Schneider ist mit einer Schreibmaschinentastatur ausgestattet, was für den Anwender recht vorteilhaft ist. Suchen Sie die deutschen Umlaute auf der Tastatur? Diese sind leider nicht fest eingebaut; allerdings kann man einzelne Zeichen umdefinieren und somit die gewünschten Zeichen erhalten.

Bei der Tastatur handelt es sich um eine sogenannte ASCII-Tastatur, die sich gut bedienen läßt. Auch die unterschiedliche Farbgebung der einzelnen Tasten ist für den Anfänger recht hilfreich, denn wichtige Tasten sind so auf den ersten Blick erkennbar.

Als nachteilig erweist sich die Tastatur des Schneiders allerdings, wenn es um das Ändern von Programmzeilen geht. Mit dieser Frage werden wir uns ein wenig später noch im einzelnen beschäftigen.

Doch nun zur Beschreibung der Tastatur.

3.1 Die Taste <ENTER>

Eine der wichtigsten Tasten des CPC 464 ist die große Taste mit der Aufschrift ENTER. Sie hat die Aufgabe, Eingaben vom Bildschirm in den Speicher des Computers zu übergeben. Alles, was Sie über die Tastatur eingeben, müssen Sie nach Abschluß der Eingabe mit der <ENTER>-Taste bestätigen.

Die <ENTER>-Taste ist mit der Zeilenvorschub-Taste auf der Schreibmaschine vergleichbar. Bei anderen Systemen heißt die <ENTER>-Taste auch <RETURN>- oder <NEW LINE>-Taste. Meistens handelt es sich dabei um die größte Taste, die sich auf der Tastatur befindet.

Wir wollen nun gleich zur Praxis übergehen, da Sie so am besten die einzelnen Tasten kennen- und verstehenlernen. Schreiben Sie einmal Ihren Namen auf dem Computer, indem Sie die einzelnen Buchstaben wie bei einer Schreibmaschine eingeben. Wenn alles geklappt hat, erscheint Ihr

Name in Kleinbuchstaben auf dem Bildschirm. Drücken Sie nun die Taste <ENTER>, um die Eingabe zu bestätigen.

Der Schneider gibt Ihnen daraufhin die Fehlermeldung *Syntax error* aus. Diese Meldung besagt, daß hier eine Eingabe vom Benutzer gemacht wurde, die der Computer nicht angenommen hat.

Mit dem Rechner kann man nur in Verbindung treten, wenn man bestimmte Befehle benutzt, die dem System bekannt sind. Ihr Name wurde vom Computer als Befehl gedeutet, den der Schneider nicht kennt. Deshalb erschien die Meldung: *Syntax error*.

Um es gleich vorweg zu nehmen: Ein solcher Fehler ist der beste, der Ihnen bei der Eingabe passieren kann. Beim "Syntax error" hat man lediglich einen Befehl nicht richtig geschrieben oder eine Folge von mehreren Befehlen nicht richtig kombiniert. Sie werden bei der Überprüfung recht schnell erkennen, wo der Fehler liegt, denn so viele Möglichkeiten gibt es meistens nicht.

Schlechter sieht es bei Fehlern aus, die an einer beliebigen Stelle im Programm auftreten können. In solchen Fällen bleibt Ihnen der Blick in das Handbuch meist nicht erspart.

Doch nun zurück zu unserem Beispiel. Nach der Fehlermeldung erscheint wieder <READY> und der quadratische Cursor. Jetzt kann erneut etwas eingegeben werden.

Im nächsten Schritt wollen wir den Vor- und Nachnamen jeweils mit Großbuchstaben beginnen lassen.

3.2 Die Taste <SHIFT>

Großbuchstaben erhalten Sie, wenn Sie die <SHIFT>-Taste und gleichzeitig den gewünschten Buchstaben drücken. Probieren Sie es einmal aus, und vergessen Sie dabei nicht, die Taste <SHIFT> festzuhalten. Jetzt sollte der erste Buchstabe Ihres Namens in Großschrift auf dem Bildschirm stehen. Mit der Taste <SHIFT> erhalten Sie also die Großbuchstaben. Sie können nun ab sofort alle Texte mit Groß- und Kleinbuchstaben eingeben, wie Sie es gerade benötigen.

Betrachten Sie jetzt einmal die obere Tastenreihe des Schneiders. Dort befinden sich über den Ziffern noch andere Zeichen. Diese Zeichen erhalten Sie ebenfalls mit der Taste <SHIFT>. Probieren Sie es einmal aus!

Ist der Versuch geglückt, können Sie wieder die Taste <ENTER> drücken. An der Fehlermeldung brauchen Sie sich nun nicht mehr zu stören, da sie Ihnen inzwischen bekannt ist.

3.3 Die Taste <CAPS LOCK>

Angenommen, Sie möchten einen längeren Text nur mit Großbuchstaben überschreiben, dann müßten Sie nach Ihren bisherigen Kenntnissen bei jedem Buchstaben die Taste <SHIFT> festhalten. Das wäre jedoch ein sehr mühsames Vorgehen!

Sehen Sie sich deshalb einmal die linke Seite der Tastatur an. Hier finden Sie vier farblich abgesetzte Tasten, von denen uns jetzt die zweite von unten besonders interessiert. Die Taste mit der Aufschrift CAPS LOCK wirkt wie ein Schalter. Wenn Sie diese Taste betätigen, werden anschließend alle Buchstaben in Großschrift ausgegeben, ohne daß Sie die <SHIFT>-Taste benutzen müssen.

Wird die <CAPS LOCK>-Taste erneut gedrückt, schaltet der Schneider wieder um, und die Großbuchstaben können wieder mit <SHIFT> erreicht werden.

Nachteilig an der <CAPS LOCK>-Taste ist, daß Sie nicht mit einem Blick erkennen können, in welchem Zustand sich diese Taste gerade befindet. Hier hätte man sinnvollerweise eine Leuchtdiode einbauen sollen, wie es bei einigen anderen Computersystemen der Fall ist. Die Leuchtdiode leuchtet immer dann, wenn <CAPS LOCK> aktiviert wurde. Aber man kann von einem so preiswerten Computersystem, das außerdem noch überdurchschnittlichen Leistungen ausgestattet ist, nicht alles verlangen!

Angenehm ist jedenfalls, daß diese Taste überhaupt vorhanden ist, denn häufig fehlen derartige arbeitserleichternde Einrichtungen bei anderen Systemen vollkommen.

3.4 Die Taste <DELETE>

Auf dem großen Tastenfeld finden Sie rechts oben die Taste <DEL>. Mit dieser Taste haben Sie die Möglichkeit, Zeichen vom Bildschirm wieder

zu löschen. Dabei wird jedes Zeichen, das links vom Cursor steht, beim Betätigen der <DEL>-Taste gelöscht. Dazu gleich ein Beispiel:

Drücken Sie zuerst wieder die <ENTER>-Taste, falls Sie dies noch nicht getan haben. Geben Sie nun irgendwelche Buchstaben oder Zahlen ein. Ihr Cursor steht danach an der letzten Stellen hinter den Zeichen.

Wenn Sie nun die Taste <DEL> betätigen, sehen Sie, wie jedesmal das letzte Zeichen von hinten gelöscht wird und der Cursor an dessen Stelle positioniert wird. Diese Möglichkeit ist sehr nützlich, wenn man sich bei der Eingabe vertippt hat und den Fehler korrigieren möchte. Wenn Sie alle Zeichen gelöscht haben und immer noch versuchen, die <DEL>-Taste zu drücken, meldet sich der Schneider mit einem Piepton.

3.5 Die Taste <CLR>

Diese Taste dient ebenfalls zum Löschen von Zeichen, allerdings werden hier diejenigen Zeichen gelöscht, die rechts vom Cursor stehen. Auch dazu ein Beispiel:

Drücken Sie wieder die Taste <ENTER> und geben Sie dann einen beliebigen Text ein. Ihr Cursor steht danach an der letzten Stelle. Wenn Sie nun versuchen, die Taste <CLR> zu betätigen, hören Sie wieder einen Piepton. Dieser Ton entsteht, weil sich rechts vom Cursor kein Zeichen befindet, das sich löschen läßt.

Auf der Tastatur finden Sie oben rechts vier Tasten mit Pfeilen. Mit diesen Pfeilen können Sie den Cursor auf eine von Ihnen gewünschte Stelle positionieren und dann Texte auf dem Bildschirm bearbeiten.

Drücken Sie jetzt einige Male die Taste mit dem Pfeil nach links. Sie sehen, wie der Cursor daraufhin nach links wandert, ohne ein Zeichen zu löschen. Jetzt wollen wir einmal alle Zeichen löschen, die sich rechts vom Cursor befinden. Dazu betätigen Sie die Taste <CLR>, was zur Folge hat, daß sich der gesamte Text beim Löschen nach vorne zieht.

Wir haben also gesehen, beide Tasten (sowohl <DEL> als auch <CLR>) löschen Zeichen. Der Unterschied besteht nur darin, ob von rechts oder von links gelöscht wird. Für den Anfang ist die Benutzung der Taste <DEL> die einfachste Möglichkeit, um Zeichen wieder vom Bildschirm zu entfernen.

3.6 Die Taste <ESCAPE>

Die <ESCAPE>-Taste unterbricht eine Ausführung des Rechners oder hält den Ablauf einer Tätigkeit an. Folgendes ist bei dieser Taste zu beachten:

Beim ersten Betätigen wird die Ausführung des Rechners angehalten, beim zweiten Betätigen wird die Ausführung unterbrochen. Hat man die Taste <ESC> nur einmal gedrückt und den Ablauf einer Ausführung angehalten, kann dieser wieder in Gang gesetzt werden, wenn anschließend eine beliebige andere Taste betätigt wird.

3.7 Der Ziffernblock

Mit dieser Bezeichnung ist der Block von Zifferntasten gemeint, der sich rechts von der Tastatur befindet. Er dient in erster Linie der schnelleren Dateneingabe von Zahlen. Allerdings gibt es auch eine andere Möglichkeit der Nutzung: Sie können nämlich aus allen diesen Tasten *Funktionstasten* machen. Funktionstasten sind Tasten, bei deren Betätigung ein bestimmter Vorgang ausgeführt wird. Ein Beispiel dazu:

Sie benötigen für eine bestimmte Bezeichnung immer wieder denselben, längeren Text. Wenn Sie sich diesen Text auf eine der Tasten legen, brauchen Sie später nur die entsprechende Taste zu drücken, und der gewünschte Text erscheint auf dem Bildschirm. Dies ist eine sehr vorteilhafte Einrichtung, die man möglichst häufig nutzen sollte.

3.8 Die Cursortasten und die <COPY>-Taste

Um es gleich vorweg zu sagen: Für mich ist die <COPY>-Taste das größte Ärgernis am gesamten System! Hier hat der Fortschritt einen Schritt zurück gemacht. Auch Sie werden bald bemerken, wie umständlich das Arbeiten mit dieser Taste auf die Dauer wird.

Zunächst möchte ich Ihnen jedoch erklären, wie man Eingaben ändert, wenn man bereits mit der Taste <ENTER> die Eingabe bestätigt hat. Die einfachste Möglichkeit wäre, mit der Cursortaste nach oben zu fahren, bis der Cursor sich auf der ersten Position der Eingabe befindet, die man ändern möchte. Viele Systeme arbeiten auf diese Weise, da es die schnellste

Korrekturmöglichkeit ist. Der Schneider wendet allerdings ein Verfahren an, von dem sich viele Hersteller schon vor längerer Zeit getrennt haben.

Versuchen Sie einmal, nachträglich einen mit <ENTER> bestätigten Namen zu verändern. Zwar läßt sich der Cursor in die Zeile bringen, in der der zu ändernde Eintrag steht; alle Versuche, hier mit <DEL> oder <CLR> zu arbeiten, werden jedoch mit dem schon bekannten Piepton abgelehnt.

Und nun zur Arbeitsweise mit dem Schneider. Drücken Sie die Taste <SHIFT> und halten Sie diese Taste fest. Gleichzeitig fahren Sie mit der Cursortaste nach oben, an den Anfang der zu ändernden Zeile. Wie Sie sehen, bleibt ein Cursor unten stehen, während der andere Cursor nach oben bis auf die zu ändernde Zeile wandert. Steht der Cursor nun am Anfang der Zeile, drücken Sie die Taste <COPY> und halten Sie diese fest, bis die gesamte obere Zeile als Kopie unten auf dem Bildschirm steht.

Achtung! Sie können jetzt nur in der Kopie Veränderungen vornehmen und müssen dann wieder mit <ENTER> bestätigen. Danach wird die Kopie der Zeile als maßgebend angesehen.

Teile einer Zeile werden kopiert, wenn Sie einzeln auf <COPY> drücken. Sie können sich sicher vorstellen, wie umständlich Änderungen in Zeilen werden, die etwas länger sind.

Bis ich mich in die gesamte Technik des Kopierens eingearbeitet hatte, verging eine ganze Weile. Seien Sie deshalb nicht ungeduldig, wenn es auf Anhieb nicht so klappt, wie es sein sollte.

Noch ein Tip: Oft geht es schneller, wenn Sie die Texte neu schreiben.

Im Verlauf der nächsten Kapitel werde ich noch darauf eingehen,wie man die Technik des Änderns einfacher handhaben kann.

3.9 Die Taste <CONTROL>

Die Taste <CONTROL> oder <CTRL> hat nur im Zusammenhang mit anderen Tasten eine Funktion. Falls Sie den Schneider neu starten möchten, kann der Einschaltvorgang mit den Tasten <ESC>, <SHIFT> und <CTRL> simuliert werden. Alle drei Tasten müssen gemeinsam betätigt werden, damit die Systemmeldung erscheint. Ich lege dafür auf der rechten Seite der Tastatur einen Finger auf die <SHIFT>- und einen auf

die <CTRL>-Taste. Mit der linken Hand drücke ich dann kurz die <ESC>-Taste und die Einschaltmeldung erscheint. Üben Sie diesen Drei-Fingergriff, Sie werden ihn noch häufiger benötigen.

Wenn Sie die Taste <CTRL> und die kleine <ENTER>-Taste im Ziffernblock gemeinsam betätigen, erscheint eine Meldung. Diese Meldung besagt, daß Sie die Taste <PLAY> am Kassettenrekorder und dann auf der Tastatur eine weitere Taste zur Bestätigung drücken sollen. Damit wird automatisch die Demodiskette geladen, auf der sich einige interessante Programme befinden. Viel Spaß bei dieser Kassette!

4 Einführung in die Programmiersprache BASIC

In diesem Kapitel möchte ich Grundsätzliches zum BASIC des Schneiders sagen, das zu einem der stärksten gehört, die ich persönlich kenne.

Für viele ist es vielleicht ganz interessant, was das Wort BASIC überhaupt bedeutet. BASIC ist die amerikanische Abkürzung für:

Beginners
All-Purpose
Symbolic
Instruction
Code

BASIC ist eine Computersprache, die es erlaubt, mit dem Rechner im Dialog zu arbeiten. Dies geschieht mit Hilfe von BASIC-Befehlen, von denen der Schneider eine ganze Menge zur Verfügung stellt. Die Kommunikation findet also über diese Schlüsselwörter statt.

Die Programmiersprache BASIC wurde in den sechziger Jahren entwickelt und hat seitdem ihren Siegeszug um die ganze Welt angetreten. Heute hat fast jeder Mikrocomputer die Programmiersprache BASIC fest eingebaut und beim Anschalten des Systems wird sie aus einem Speicher geladen.

Leider bauen die Hersteller in ihre BASIC-Versionen im allgemeinen Befehle ein, die auf anderen Computern nicht laufen. Nur die einfachsten Grundbefehle sind bei allen Computersystemen gleich. Bei speziellen Einrichtungen, wie beispielsweise Grafiken oder Datenspeicherung, gehen dann die Wege auseinander. Deshalb existieren heute viele BASIC-Versionen, die von System zu System unterschiedlich sind.

Beherrscht man aber die wichtigsten Befehle und kennt sich mit der Arbeitsweise von BASIC aus, kann man schnell auf eine der anderen Versionen umsteigen.

4.1 Der Direktmodus

BASIC erlaubt die Eingabe von Befehlen, die direkt über die Tastatur erfaßt werden können. Diese Befehle werden - wie bei einer Schreibmaschine - Zeichen für Zeichen eingetippt. Im Direktmodus eingegebene Befehle führt der Schneider nach Betätigen der <ENTER>-Taste sofort aus. Diese Befehle können in Groß- oder Kleinschrift eingegeben werden. Ein sehr wichtiger Befehl, den wir später noch ausführlicher besprechen wollen, soll uns dafür zunächst als Beispiel dienen.

Es handelt sich dabei um den <PRINT>-Befehl, der zur Ausgabe von Zahlen und Zeichen auf dem Bildschirm benötigt wird. Bitte geben Sie nun ein:

```
PRINT 10 + 10
```

Drücken Sie jetzt zur Bestätigung der Eingabe die Taste <ENTER>. Auf dem Bildschirm erscheint daraufhin das Ergebnis, nämlich eine 20.

Mit diesem Befehl können auch Texte auf dem Bildschirm ausgegeben werden. Möchte man z.B. seinen Namen sehen, gibt man ihn mit zwei Anführungsstrichen ein, wie etwa: `PRINT"Anton Schneider"`. Nach dem Drücken der Taste <ENTER> erscheint der Name Anton Schneider; der Befehl `PRINT` und die beiden Anführungsstriche sind jedoch verschwunden. Sie erhalten also nur den gewünschten Namen als Ausgabe.

Der häufigste Fall, bei dem man im Direktmodus arbeitet, ist die Benutzung des Computers als Taschenrechner. Der Schneider beherrscht alle Grundrechenarten, die ich nachfolgend an einigen Beispielen demonstrieren möchte. Die Beispiele haben eines gemeinsam: Alle Eingaben erfolgen im Direktmodus und werden nach Betätigen der <ENTER>-Taste sofort ausgeführt. Bevor wir beginnen, kurz noch die Erläuterung der Bedeutung der einzelnen mathematischen Symbole.

+	entspricht der Addition
-	entspricht der Subtraktion
*	entspricht der Multiplikation
/	entspricht der Division
^	entspricht der Potenzierung
.	entspricht dem Dezimalpunkt
(	entspricht der Klammer auf
)	entspricht der Klammer zu

Beispiel 1:

Eine einfache Addition soll ausgeführt werden (123 + 34).

Der dazu nötige Befehl lautet:

```
PRINT 123 + 34
```

Als Ergebnis dieser Aufgabe erscheint der Wert 157 auf dem Bildschirm.

Beispiel 2:

Eine einfache Addition mit Nachkommastellen soll ausgeführt werden. Die beiden Werte 1.34 und 0.15 sollen miteinander addiert werden. Dies geschieht mit dem folgenden Befehl:

```
PRINT 1.34 + 0.15
```

Als Ergebnis erscheint 1.5 und nicht 1.50, da die letzte Null in einem solchen Fall immer unterdrückt wird.

Beispiel 3:

Eine gemischte Rechnung mit Subtraktion und Addition soll durchgeführt werden. Die Werte 100 und 155 sollen addiert und vom Ergebnis anschließend 25 subtrahiert werden. Diese Rechnung lösen Sie wieder mit einem PRINT-Befehl, der folgendermaßen aussieht:

```
PRINT 100+155-25
```

Vergessen Sie die <ENTER>-Taste nicht. Das Ergebnis muß jetzt auf dem Bildschirm sichtbar sein.

Beispiel 4:

Die Fläche eines Kreises soll berechnet werden, wobei der Durchmesser bekannt ist. Angenommen, der Durchmesser ist 24.78, dann sieht der PRINT-Befehl wie folgt aus:

```
PRINT 24.78*24.78*PI/4
```

Beachten Sie bitte die Konstante PI. Der Schneider hat unter dieser Bezeichnung den Wert von 3.141 fest gespeichert.

Beispiel 5:

Eine einfache Klammerrechnung soll gelöst werden. Eine Beispielrechnung könnte dabei so aussehen:

```
PRINT 2.67*(45-67+3)
```

Das Ergebnis erhalten Sie wieder nach Betätigung der <ENTER>-Taste.

4.2 Der Programm-Modus

Damit der Computer etwas sofort ausführt, wird der Direktmodus verwendet. Im letzten Kapitel lernten wir diese Arbeitsweise bereits kennen. Nun gibt es aber auch die Möglichkeit, Befehle zu sammeln und diese dann bei einem bestimmten Schlüsselwort ausführen zu lassen. Dazu wird zunächst am Anfang der Zeile eine Zahl, die sogenannte *Zeilennummer* eingegeben, die die Reihenfolge bestimmt, in der die Befehle ausgeführt werden. Der Computer beginnt mit der Ausführung seiner Arbeit dann bei der Zeile mit der niedrigsten Zeilennummer und führt diese Zeile aus. Das folgende Beispiel soll dies verdeutlichen:

```
10    PRINT"Das ist Zeile 10"
20    PRINT"Das ist Zeile 20"
30    PRINT"Hier ist Schluss"
```

Wenn Sie dieses kleine Programm mit dem Befehl RUN starten, sehen Sie auf dem Bildschirm folgendes:

```
Das ist die Zeile 10
Das ist die Zeile 20
Hier ist Schluss
```

Das Programm hat Zeile 10 zuerst und Zeile 30 zuletzt ausgeführt. Der Befehl PRINT wird in den nächsten Kapiteln eingehender beschrieben und diente hier nur zur Demonstration.

Ein Doppelpunkt innerhalb einer Zeile trennt einzelne Befehle voneinander und ermöglicht in einer Zeilennummer mehrere BASIC-Befehle.

Beispiel:

```
10 PRINT "Das": PRINT "ist": PRINT "die Zeile 10"
```

Die Zeilennummer kann theoretisch bei 1 beginnen und bis in den Bereich von 63000 gehen. Sie sollten jedoch möglichst gerade Zeilennum-

mern verwenden und eine Schrittweite von 10 einhalten, also z.B.: 10, 20, 30.

Dieses Vorgehen hat den Vorteil, daß Sie später zwischen 10 und 20 noch weitere Zeilen einfügen können, was bei einer Folge von 1, 2, 3 usw. nicht möglich ist.Eine Zeilennummer, die 1.5 lautet, gibt es im BASIC nicht. Deshalb die großen Abstände zwischen den Zeilennummern.

4.3 Variablen und Konstanten

Ein Computerprogramm ist eine Zusammensetzung aus Zeilennummern, Befehlen und Variablen. Der eigentliche Ablauf eines Programms ändert sich dabei niemals, sondern nur die Daten, die eingegeben oder berechnet werden. Zum besseren Verständnis dazu ein kleines Beispiel:

```
10    PRINT 10+1
20    PRINT 23-2
30    PRINT 45/5
```

Dieses Programm rechnet Ihnen zwar die angegebenen Rechenbeispiele aus, ist jedoch in keiner Weise variabel. Im Prinzip kann man ein solches Programm nicht sinnvoll nutzen, da kein Einfluß auf die jeweiligen Werte in der Rechenoperation genommen werden kann.

Ein Weg, dieses Problem zu lösen, sieht folgendermaßen aus: Stellvertretend für Zahlen und Texte wird mit Buchstaben gearbeitet, die einen bestimmten Inhalt haben. Diese Buchstaben nennt man *Variablen* und *Konstanten*. Doch darüber später mehr. Erst möchte ich Ihnen zeigen, wie ein Computer seine Daten speichern kann.

Alle Eingaben und Ergebnisse, die ein Computer verarbeitet, muß er irgendwo speichern, sonst sind sie verloren. Damit der Computer weiß, wo sich bestimmte Daten im Speicher befinden, vergibt man für diesen speziellen Speicherplatz einen Namen. Man kann das mit einem großen Aktenschrank mit vielen Schubladen vergleichen, von denen jede einen unverwechselbaren Namen hat. Möchten Sie z.B. in die Schublade mit dem Namen A eine ganz bestimmte Zahl (vielleicht die 50) legen, dann können Sie in der Programmiersprache BASIC folgende Zeile schreiben:

```
Zeilennummer A=50
```

Vergessen Sie nicht, die Taste <ENTER> zu drücken, damit die Eingabe auch vom Rechner übernommen wird.

Sie haben jetzt in Schublade A den Wert 50 gelegt. Dieser Wert steht für weitere Bearbeitungen in der Schublade A zur Verfügung.

Hat eine Schublade von Anfang an einen bestimmten Inhalt, den Sie festgelegt haben, spricht man von einer sogenannten *Konstanten.* Wenn Sie beispielsweise einen Briefkopf schreiben möchten, werden Sie die Daten dafür in Konstanten ablegen und sie im weiteren Verlauf des Programms wieder verarbeiten.

Kommen erst später Daten in die Schubladen, z.B. durch Eingaben oder Berechnungen, spricht man von Variablen. *Variablen* ändern ständig ihren Inhalt, wenn Sie z.B. beim erneuten Start des Programms andere Werte eingeben. Variablen können numerische oder alphanumerische Daten speichern. Für Konstanten trifft dies ebenfalls zu. Das folgende Schema soll dies noch einmal verdeutlichen.

Zusammenfassung:

Hat eine Variable von Anfang an einen fest vorbestimmten Inhalt, dann handelt es sich um eine Konstante. In allen anderen Fällen spricht man von Variablen, da sich der Inhalt im Verlauf des Programms ändern kann. Variablen und Konstanten verhalten sich von der Be- und Verarbeitung her gesehen gleich.

4.4 Variablennamen

Wie wir bereits gesagt haben, kann man in verschiedene Schubladen (Variablen) beliebige Daten speichern. In diesem Abschnitt werden wir uns mit der Bezeichnung der Schubladen beschäftigen, die später einen bestimmten Inhalt aufnehmen sollen.

Während andere vergleichbare Computersysteme dieser Preisklasse nur Variablen bis zu zwei Zeichen Länge verarbeiten können, bietet Ihnen der Schneider hier wesentlich mehr an. Sie können beliebige Variablennamen benutzen, wenn Sie die folgenden zwei Bedingungen beachten:

a) Der Name der Variablen darf eine Gesamtlänge von 40 Zeichen nicht überschreiten.

b) Sie dürfen keine Schlüsselwörter verwenden, die auch der Schneider benutzt.

Schlüsselwörter sind BASIC-Befehle, die der Rechner für seine Arbeit benötigt. Diese Schlüsselwörter dürfen Sie also nicht benutzen, da es sonst im späteren Programmablauf zu Fehlern kommt.

Grundsätzlich lassen sich sowohl Zahlen als auch Texte speichern. Bei den Zahlen spricht man in der Fachsprache von numerischen Daten und bei den Texten von sogenannten alphanumerischen Daten.

Da ein Computer diese beiden Typen nicht ohne weiteres auseinanderhalten kann, muß man ihm vorher sagen, in welchem Variablentyp man bestimmte Dinge speichern möchte. Dabei gilt folgendes:

Alphanumerische Variablen bekommen immer ein Dollarzeichen am Ende des Variablennamens angehängt ($). Bei *numerischen Variablen* entfällt dieses Dollarzeichen. Wird dies nicht beachtet, kommt es zwangsläufig zu Fehlern:

Variablennamen können gleichzeitig Zeichen und Zahlen enthalten. Wer z.B. seine Umsätze der letzten drei Jahre in Variablen abspeichern möchte, kann folgende Bezeichnung wählen:

```
UMSATZ1983
UMSATZ1984
UMSATZ1985
```

Wichtig ist dabei vor allem, daß keine Leerräume zwischen den Buchstaben und den Zahlen stehen. Das gilt grundsätzlich für alle Variablennamen. Also keine Leerzeichen innerhalb von Variablennamen verwenden!

Weiter oben wurden die Schlüsselwörter angesprochen und dazu gesagt, daß diese nicht als Variablen verwendet werden dürfen. Sie können jedoch Schlüsselwörter in Variablennamen einbauen. So enthält beispielsweise die Variable ENDE den Schlüsselbegriff END. Ein weiteres Beispiel ist die Variable PRINTERON, die das Schlüsselwort PRINT enthält. Derartige Variablenbezeichnungen sind zulässig. Es spielt keine Rolle, ob sie die Variablenbezeichnungen groß oder klein schreiben.

Nachfolgend möchte ich Ihnen eine Reihe von Variablen vorstellen, die von ihrer Schreibweise her nicht ganz in Ordnung sind.

Falsch	**Richtig**
1A	A1 - Zahlen dürfen niemals vorne stehen
A 1	A1 - keine Leerräume innerhalb einer Variable
TIME	entfällt, da ein BASIC-Schlüsselwort
WERT!	WERT - ein solches Zeichen ist nicht zulässig
TEXT$1	TEXT1$ - das Dollarzeichen muß hinten stehen
F/	F - Schrägstrich nicht erlaubt
ANZAHL-ZEICHEN	ANZAHLZEICHEN - Bindestrich unzulässig
ZAHL&	ZAHL - unzulässiges Zeichen benutzt
NAME%$	NAME$ - unzulässiges Zeichen benutzt

4.5 Tips für die Vergabe von Variablennamen

Sie haben in der Wahl eines Variablennamens freie Hand und müssen dabei nur aufpassen, daß Sie 40-Zeichen-Länge nicht überschreiten und kein BASIC-Schlüsselwort verwenden. Wählen Sie für alle Programmentwicklungen möglichst aussagekräftige Variablennamen, damit Sie Ihre Programme später noch gut lesen und verstehen können.

Was damit gemeint ist, soll an zwei Beispielen aufgezeigt werden.Das erste Beispiel ist von der Variablenwahl her nicht sonderlich gut. Das Programm berechnet das Volumen aus den Angaben Länge, Breite und Höhe.

```
10    L=100
20    B=123
30    H=2
40    V=L*B*H
50    PRINT V
```

Bedeutend verständlicher ist da schon das folgende Programm, das dem gleichen Zweck dient:

```
10    LAENGE=100
20    BREITE=123
30    HOEHE=2
40    VOLUMEN=LAENGE*BREITE*HOEHE
50    PRINT VOLUMEN
```

Auf den ersten Blick erkennt auch ein ungeübter Beobachter, was hier berechnet wird. Besonders längere Programme werden durch die Wahl von aussagekräftigen Variablen wesentlich übersichtlicher. Wählen Sie daher am besten Namen, die auch Rückschlüsse auf den Inhalt der Variablen zulassen. Die nun folgende Übersicht soll Ihnen dabei behilflich sein.

Sie wollen speichern	Vorschläge für Variablen
Zahlen	ZAHL
Vornamen	VORNAME$
Wohnorte	ORT$
Umsätze	UMSATZ
Rechnungsdatum	DATUM$
Mittelwerte	MITTELWERT
Gewichte	GEWICHT
Artikelnummern	ARTNR
Artikelbezeichnungen	ARTBEZ$
Zählvorgänge	ANZAHL

Sie sehen, die richtige Wahl der Variablennamen ist eigentlich gar nicht so schwer, wie es auf den ersten Blick scheint.

4.6 Erste Versuche mit Variablen und Konstanten

Stellen Sie sich einmal vor, Sie benötigen innerhalb eines Programms zur Erstellung eines Briefkopfes Ihre eigene Anschrift. Diese Anschrift müßten Sie jedesmal, wenn Sie ein Programm mit RUN starten, komplett neu eingeben. Da das auf die Dauer recht zeitraubend ist, können Sie den Computer mit dieser Aufgabe beauftragen. Hier gibt es nun zwei Möglichkeiten, die beide ans Ziel führen.

Möglichkeit 1:

Sie schreiben ein Programm, daß mit Hilfe von PRINT-Befehlen Ihre komplette Anschrift am Anfang auf dem Bidlschirm ausgibt.

```
10    PRINT"Anton Schneider"
20    PRINT"Beerenweg 13"
30    PRINT"4875 Irgendwo 11"
40    PRINT"Telefon 09871/5454"
```

Starten Sie später Ihr Programm, dann erscheint jedesmal am Anfang Ihre Anschrift auf dem Bildschirm, ohne daß Sie weitere Eingaben vornehmen müssen. Sie brauchen in diesem Fall die Anschrift nur einmal programmieren. Mit einem entsprechenden Zusatz hinter dem PRINT-Befehl kann sie später auf einem Drucker ausgegeben werden.

Möglichkeit 2:

Dieser Lösungsweg ist etwas anders und arbeitet mit den schon besprochenen Konstanten. Konstanten sind bekanntlich Variablen, die einen festen Inhalt haben, der Ihnen am Programmanfang zugewiesen wird. Diese Zuweisung erspart ebenfalls die wiederholte Eingabe der Anschrift, hat aber zusätzlich noch einen weiteren Vorteil. Sie können nämlich, sooft es erforderlich ist und an jeder Stelle des Programms, die Anschrift, oder auch nur Teile davon, aufrufen. Bei der zuerst aufgeführten Lösung war dies nur am Anfang des Programms möglich und dann auch nur ein einziges Mal. Nachfolgend nun das entsprechende Programm zum zweiten Lösungsweg:

```
10    NAME$="Anton Schneider"
20    STRASSE$="Beerenweg 13"
30    ORT$="4875 Irgendwo 11"
40    TELEFON$="Telefon 09871/5454"
```

Starten Sie jetzt das Programm mit **RUN** und betrachten Sie sich das Ergebnis. Sie werden sicherlich außer einem **READY** nichts weiter sehen.

Da in den vier Programmzeilen kein **PRINT**-Befehl stand, wurde auch nichts auf dem Bildschirm ausgegeben. Dennoch hat das Programm seinen Zweck erfüllt; es wurde ja nur damit beauftragt, den vier Konstanten einen festen Inhalt zuzuweisen und nicht damit, eine Bildschirmausgabe zu machen.

Um sich davon zu überzeugen, geben Sie bitte im Direktmodus, also ohne Zeilennummer, ein:

```
PRINT NAME$
```

Wenn Sie die <ENTER>-Taste gedrückt haben, erscheint der Name "Anton Schneider" auf dem Bildschirm. Er ist jederzeit erneut abrufbar. Entsprechend können Sie mit den restlichen drei Konstanten verfahren und sich davon überzeugen, daß auch sie ihren vorbestimmten Inhalt haben.

Beachten Sie bitte die Zuweisung der Konstanten. Texte werden immer mit Anführungsstrichen zugewiesen, Zahlen dagegen ohne diese Zeichen!

Wollen Sie beispielsweise Ihren Brief mit den Worten "Mit freundlichen Grüßen Anton Schneider" beenden, dann könnte eine programmtechnische Lösung so aussehen:

```
200   PRINT"Mit freundlichen Grüßen"
210   PRINT
220   PRINT NAME$
```

Zeile 200 liefert dabei einen Standardtext, Zeile 210 eine Leerzeile und Zeile 220 den Namen des Briefschreibers.

4.7 Variablenkennzeichnung

Oft werden in einem Programm nur Variablen benötigt, in denen später Zahlen ohne Nachkommastellen verarbeitet werden. Solche Zahlen, die man auch als Integerzahlen bezeichnet, verbrauchen im Computer weniger Speicherplatz, wenn man die Variablen vorher festlegt, in denen diese Zahlen verarbeitet werden sollen.

Beim Einschalten des Systems haben alle Variablen die Möglichkeit, Realzahlen zu speichern. Realzahlen sind Zahlen mit Vorzeichen und Nachkommastellen.

Beispiel:

12.575 ist eine Realzahl
12 ist ein Integerwert

Festlegen können Sie Ihre Variablen mit folgenden Kennzeichen:

% am Ende einer Variablen deklariert eine *Integervariable*
! deklariert eine Variable als *Realzahlvariable*

Beispiel:

A% dient zur Verarbeitung von Integerzahlen
P! dient zur Verarbeitung von Realzahlen

Hinweis: Das Ausrufungszeichen (!) kann normalerweise entfallen, weil ja das System bereits beim Einschalten diesen Variablentyp automatisch deklariert.

Möchte man z.B. ausschließlich Integerwerte verarbeiten, dann braucht man nicht hinter jede Variable ein %-Zeichen zu setzen. Auch dazu gibt es einen speziellen Befehl:

```
DEFINT
```

Nach dem Befehl folgen diejenigen Variablen, in denen nur Integerwerte verarbeitet werden sollen.

Beispiel:

```
DEFINT A - Z
```

macht alle Variablen, die mit diesen Buchstaben beginnen, zu Integervariablen. Es ist jedoch auch möglich, einzelne Bereiche anzusprechen, wie etwa mit dem Befehl:

```
DEFINT A, G-H, T, O, X-Z
```

5 Die wichtigsten BASIC-Befehle und ihre Anwendung

In diesem Kapitel finden Sie die wichtigsten BASIC-Befehle für die Ein- und Ausgaben sowie für die Verarbeitungen von Daten. Sie sollten das Kapitel besonders gründlich durcharbeiten, da es den Grundstock für alle weiteren Programmbeispiele bildet.

Experimentieren Sie mit den Programmen, bis Sie sich in der Anwendung der Befehle sicher fühlen. Versuchen Sie es auch einmal mit kleinen Abwandlungen und sehen Sie sich das Ergebnis an. Auf diese Weise lernen Sie am besten die Programmierung des Rechners kennen und haben später bei anderen Programmen keine Verständnisschwierigkeiten mehr.

Um eine möglichst optimale Übersichtlichkeit zu erhalten, wurde bei den Beispielen im allgemeinen folgende Reihenfolge eingehalten:

1. **Aufgabe**: Was soll gemacht werden, um was geht es dabei?
2. **Eingabe**: Welche Eingaben braucht das Programm?
3. **Verarbeitung**: Wie werden die Daten verarbeitet, damit man ans Ziel kommt?
4. **Ausgabe**: Was soll am Schluß auf dem Bildschirm erscheinen?
5. **Das Programm**: So sieht das BASIC-Programm aus.
6. **Beschreibung**: So funktioniert das Programm.
7. **Hinweise und Bemerkungen**: Was es sonst noch bei diesem Programm zu beachten gibt.

Sollten Sie Schwierigkeiten bei der Anwendung dieses Kapitels haben, liegt die Ursache vermutlich darin, daß Ihnen der Umgang mit dem Rechner noch nicht sehr vertraut ist. Aber lassen Sie sich nicht entmutigen, aller Anfang ist schwer! Wir wollen nun gleich mit den wichtigsten Befehlen beginnen.

5.1 Der Befehl CLS

Mit CLS können Sie den Bildschirm komplett löschen. "Löschen" bedeutet in diesem Fall, daß alle Daten zwar vom Bildschirm verschwinden, daß aber das Programm und die Daten noch im Speicher des Computers vorhanden sind.

Leider besitzt der Schneider keine Taste, mit der man den Bildschirm löschen kann. Zu diesem Zweck können Sie jedoch eine Funktionstaste definieren. Wie das genau gehandhabt wird, wird später noch besprochen.

Den Befehl CLS können Sie ohne jeden weiteren Zusatz anwenden.

5.2 Der Befehl RUN

Einer der wichtigsten Befehle heißt RUN. Er dient dazu, nach Betätigung der <ENTER>-Taste ein BASIC-Programm zu starten. Das Programm beginnt mit seiner Ausführung bei der niedrigsten Zeilennummer, die es vorfindet.

Achtung! Beachten Sie immer, daß alle eingegebenen Daten mit RUN wieder gelöscht werden.

Mit dem RUN-Befehl können Sie außerdem den Sofortstart eines Programms direkt von Kassette oder Diskette bewirken. Sie brauchen dazu nur den Namen des Programmes hinter "RUN" einzugeben.

Beispiel: Das Programm mit dem Namen "flasche" soll geladen werden und sich dann von selbst starten. Geben Sie dazu nur den Namen an und schon geht es los:

```
RUN"flasche"
```

Nach Betätigung der Taste <ENTER> lädt sich der Rechner das Programm und startet es von selbst.

5.3 Der Befehl LIST

Mit diesem Befehl können Sie sich Ihre Programme auf dem Bildschirm ansehen. Der Befehl ist folgendermaßen anzuwenden:

LIST listet alle Zeilennummern auf dem Bildschirm

LIST-20 listet alle Zeilennummern bis Zeile 20

LIST 50 listet nur Zeile 50

LIST 30-90 listet Zeile 30 bis 90

LIST 50- listet die Zeilennummern ab Zeile 50 bis zum Ende des Programms.

5.4 Der Befehl REM

Dieser Befehl wird Ihnen immer wieder im vorliegenden Buch begegnen. Mit dem REM-Befehl können Sie beliebig viele Texteinschübe und Hinweise in ein Programm einfügen, die es lesbarer machen.

REM-Zeilen sind nur im BASIC-Listing sichtbar, im Programm haben sie später keine Bedeutung. Wenn der Computer ein Programm abarbeitet und auf einen solchen REM-Befehl trifft, springt er zur nächsten Zeilennummer weiter.

Sie sollten sich in der ersten Zeit ruhig angewöhnen, diesen Befehl recht häufig einzusetzen. Wenn Sie sich nach einigen Wochen Ihr Programm wieder betrachten, können Sie sofort sagen, welche Zeilen für was zuständig sind. Die Lesbarkeit eines Programms wird dadurch deutlich erhöht, und auch ein Fremder kann leichter mit Ihrem Programm arbeiten, wenn es gut dokumentiert ist.

Das Programm, das nun folgt, bewirkt im Prinzip überhaupt nichts.

```
10    REM **********************************************************
20    REM ***  Dieses Programm macht nichts, da es nur aus      ***
30    REM ***  REM-Zeichen besteht                              ***
40    REM ***                                                   ***
50    REM *** Auch Copyright und Programm-Namen können hier     ***
60    REM *** hinter den REM-Zeilen stehen                      ***
70    REM **********************************************************
80    REM
90    REM das wär's
```

Leider haben REM-Befehle auch einen Nachteil: Sie verbrauchen Speicherplatz, der an anderer Stelle vielleicht benötigt wird. Wenn der Speicherplatz also einmal knapp werden sollte, trennen Sie sich zuerst von den REM-Zeilen.

Als Abkürzung für diesen Befehl kann das Hochkomma (') verwendet werden.

Achtung: In einer REM-Zeile darf der senkrechte Balken (<SHIFT +>) nicht verwendet werden.

5.5 Der Befehl PRINT

Dieser Befehl dient zur Datenausgabe auf dem Bildschirm. Alles, was auf dem Bildschirm erscheinen soll, muß vorher mit dem Befehl PRINT gekennzeichnet werden. Zu den verschiedenen Kombinationsmöglichkeiten des PRINT-Befehls werden nachfolgend einige Beispiele aufgeführt.

Beispiel 1:

```
10    PRINT
20    PRINT
30    PRINT
```

Wenn Sie dieses Programm mit RUN starten, erhalten Sie drei Leerzeilen auf dem Bildschirm. Wird ein PRINT-Befehl ohne Zusatz verwendet, ergibt dies später auf dem Bildschirm eine Zeile, in der nichts steht. Solche Leerzeilen sind recht nützlich, wenn man einen übersichtlichen Bildschirmaufbau wünscht.

Beispiel 2:

```
10    PRINT 12
20    PRINT 8.91
30    PRINT 66.99
```

Zahlen kann man ausgeben, indem man hinter den PRINT-Befehl die gewünschten Zahlen setzt. Damit können Sie zwar noch nicht rechnen, aber Sie erhalten eine Bildschirmausgabe. Rechnen wollen wir im nächsten Schritt.

Beispiel 3:

```
10    PRINT 12 + 12
20    PRINT 144 / 12
30    PRINT 5.2 * 2
```

Dieses Beispiel führt Ihnen vor, wie Sie mit Werten, die hinter einem PRINT stehen, rechnen können.

Bisher haben wir uns nur mit Zahlen beschäftigt; aber auch Texte lassen sich mit dem Befehl PRINT ausgeben. Dabei ist allerdings ein kleiner Unterschied zu beachten: Texte müssen *immer in Anführungszeichen* stehen, wenn sie mit PRINT ausgegeben werden!

Beispiel 4:

```
10    PRINT"Das ist ein kleines Textdemo"
20    PRINT"für den SCHNEIDER CPC 464"
30    PRINT"---------------------------"
40    PRINT
50    PRINT"Auch beliebige andere Zeichen"
60    PRINT"lassen sich mit PRINT ausgeben"
70    PRINT"! # $ % & ' ( ) - * + / = . : "
80    PRINT
90    PRINT"Ende des Programms erreicht"
```

Bei den einzelnen Textzeilen können Sie die hinteren Anführungszeichen weglassen, wenn danach nichts mehr folgt. Ich empfehle dennoch, sie zu setzen; wenn Sie später vielleicht auch an anderen Computern arbeiten, fällt Ihnen der Einstieg dort leichter, wenn Sie an die Standardschreibweise gewöhnt sind.

Sie haben nun gesehen, wie Zahlen und Texte ausgegeben werden können. Als nächstes möchte ich mit Ihnen die Kombination von Zahlen und Texten innerhalb einer Zeile besprechen und sie anhand einiger Beispiele verdeutlichen. Außerdem geht es um die Verwendung des Kommas und des Semikolons bei PRINT-Zeilen.

Angenommen, Sie möchten innerhalb einer Zeile später den Schlußsatz sehen:

```
Das Ergebnis beträgt ... Quadratmillimeter
```

Anstelle des Platzhalters stehen dann natürlich richtige Werte. Wenn wir uns den Schlußsatz genau ansehen, können wir ihn in drei Teile unterteilen. Bei dem ersten Teil handelt es sich um einen Text, beim zweiten Teil um eine Zahl und beim dritten Teil wieder um einen Text. Durch Aneinanderfügen der einzelnen Teile ergibt sich dann der Schlußsatz. Und so sieht die Befehlszeile aus:

```
10    PRINT"Das Ergebnis beträgt";343;"Quadratmillimeter"
```

Das Zusammenfassen geschieht also hier mit einem Semikolon. Das Semikolon hat jedoch noch eine andere Bedeutung. Wenn es am Ende

einer PRINT-Zeile steht, wird der Zeilenvorschub in die nächste Zeile unterdrückt und beim nächsten Aufruf eines PRINT-Befehls wird an der letzten Position weitergeschrieben. Unser Beispiel hätte auch so aussehen können:

```
10    PRINT"Das Ergebnis beträgt";
20    PRINT 343;
30    PRINT"Quadratmillimeter"
```

Sie sollten sich also merken, daß ein Semikolon den Zeilenvorschub unterdrückt.

Ein weiteres Zeichen, das ebenfalls den Zeilenvorschub unterdrückt, ist das Komma innerhalb oder am Ende einer PRINT-Zeile. Das Komma als Trennzeichen einer Zeile bewirkt beim Ausdruck einen größeren Zwischenraum zwischen den einzelnen Zahlen oder Texten. Sehen Sie sich dazu einmal das folgende Beispiel an:

```
10    MODE 2
20    PRINT 1,2,3,4,5,6,7,8,9,0
```

Sie werden nach dem Programmierstart alle Zahlen wieder sehen, aber mit einem Abstand von 12 Leerzeichen. Das bedeutet, daß alle 13 Positionen eine neue Zahl auf dem Bildschirm erscheint. Solche Kommas finden z.B. Einsatz beim Aufbau von Tabellen und anderen formatierten Datenausgaben.

Standardmäßig ist also ein Abstand von 13 Zeichen eingestellt. Das bedeutet für Sie als Programmierer, 13 Zeichen später erscheint das nächste Zeichen auf dem Bildschirm.

Oft reicht aber der Abstand nicht aus oder ist viel zu kurz. Zum Verändern von Abständen gibt es einen speziellen Befehl:

ZONE

Im folgenden Beispiel wird der Befehl ZONE am praktischen Anwendungsfall demonstriert.

```
 10   MODE 2
 20   CLS
 30   PRINT"Normalsmodus (ZONE 13)"
 40   PRINT 1,2,3,4,5,6,7,8,9,0
 50   PRINT:PRINT
 60   PRINT"Jetzt wird mit ZONE 2 gearbeitet"
 70   ZONE 2
 80   PRINT 1,2,3,4,5,6,7,8,9,0
 90   PRINT:PRINT
100   PRINT"Und nun kommt ZONE 8"
110   ZONE 8
120   PRINT 1,2,3,4,5,6,7,8,9,0
```

Sie können also mit dem Befehl ZONE die Abstände der einzelnen Ausgaben beeinflussen. Vergleichen kann man diesen Befehl mit TAB, nur mit dem Unterschied, daß mit TAB jede einzelne Position angesteuert werden kann.

Zur Positionierung von Daten auf dem Bildschirm stellt das BASIC des Schneiders einen weiteren Befehl zur Verfügung. Dieser neue Befehl lautet

TAB

Er funktioniert wie der Tabulator einer Schreibmaschine. Der Tabulator wird mit PRINT TAB(22) z.B. auf die 22. Position der aktuellen Bildschirmzeile gesetzt. Dort kann man mit einem weiteren PRINT-Befehl beliebige Daten ausgeben. Dazu einige Beispiele:

```
10    MODE 2
20    CLS
30    PRINT TAB(35);"TESTPROGRAMM"
40    PRINT TAB(30);"========================"
50    PRINT
60    PRINT"Name"; TAB (20); "Vorname"; TAB (40); "Vereinseintritt am"
70    PRINT"-------------------------------------------------"
80    PRINT"Weber";TAB(20);"Albert"";TAB(40);"21.6.1974"
90    PRINT"Meier";TAB(20);"Klaus";TAB(40);"1.1.1956"
100   PRINT
110   PRINT TAB(40);"Programm-Ende"
```

Mit dem Befehl TAB wird also eine bestimmte Position innerhalb einer Bildschirmzeile angesprochen. Danach kommt die Mitteilung, die auf dieser Position ausgegeben werden soll.

Die helle Taste mit der Aufschrift TAB, links auf der Tastatur, hat mit dem Befehl TAB nichts zu tun. Sie müssen also den Befehl, wie alle anderen Befehle, als Zeichenfolge über die Tastatur eintippen.

Sie haben nun fast alle Formatierungsmöglichkeiten im Zusammenhang mit dem PRINT-Befehl kennengelernt. Es gibt noch einige weitere Befehle, mit denen wir uns im Kapitel: Formatierte Datenausgabe beschäftigen werden.

Im nächsten Abschnitt werden wir die verschiedenen Eingabemöglichkeiten besprechen, die es unter dem BASIC des CPC 464 gibt.

5.6 Der Befehl INPUT

Ein Computer, in den man keine Daten eingeben kann, ist im Prinzip wertlos. Es muß also eine Einrichtung geben, die es ermöglicht, mit dem Rechner zu kommunizieren.

Für die Eingabe von Daten stellt das BASIC einige Befehle zur Verfügung, die je nach Anwendungsfall anzuwenden sind. Der wichtigste Befehl zur Dateneingabe von Texten und Zahlen ist der Befehl INPUT. Diesen Befehl können Sie folgendermaßen anwenden:

Für Zahlen:

```
INPUT numerische Variable
```

Für Texte und Zeichen

```
INPUT alphanumerische.
```

Alle Eingaben werden in Variablen gespeichert und stehen somit zur weiteren Bearbeitung zur Verfügung.

Mit dem folgenden einfachen Beispiel wird die Fläche eines beliebigen Rechtecks berechnet.

```
10    INPUT LAENGE
20    INPUT BREITE
30    FLAECHE = LAENGE * BREITE
40    PRINT FLAECHE
```

Wenn Sie das Programm mit RUN gestartet haben, erscheint als erstes ein Fragezeichen auf dem Bildschirm. Das Programm wartet bei jedem INPUT-Befehl auf eine Eingabe, die anschließend immer mit der <ENTER>-Taste bestätigt werden muß, damit sie übernommen wird.

Wenn Sie in einem Programm viele INPUT-Anweisungen haben, erscheinen auf dem Bildschirm auch entsprechend viele Fragezeichen. Es ist leicht vorstellbar, daß Sie hierbei die Übersicht verlieren, d.h. ein Fragezeichen nicht mehr eindeutig zu der gewünschten INPUT-Anweisung zuordnen können. Daher ist es sinnvoll, das Fragezeichen mit einer Bezeichnung zu versehen, die der gewünschten INPUT-Anweisung eintspricht. Dies ist auf folgende zwei Arten möglich:

a) Sie verwenden vor dem INPUT-Befehl einen PRINT-Befehl mit einem entsprechenden Hinweis darauf, was nun eingegeben werden soll.

Beispiel:

```
10    PRINT"Bitte geben Sie die Länge ein"
20    INPUT LAENGE
30    PRINT"Bitte jetzt die Breite eingeben"
40    INPUT BREITE
```

Besonders bei sehr langen Hinweisen ist diese Möglichkeit sehr nützlich.

b) Sie verwenden einen Text in dem Befehl INPUT selbst. Meist entscheidet man sich für diese Möglichkeit, da sie für kurze Hinweise besser ist, als die oben genannte Version. Sie sparen dadurch einmal Speicherplatz und erhalten zum anderen eine größere Übersichtlichkeit.

Beispiel:

```
10    INPUT"Bitte Länge eingeben";LAENGE
20    INPUT"Breite eingeben";BREITE
```

Wenn Sie auf das Fragezeichen bei dem Befehl INPUT verzichten möchten, dann verwenden Sie anstelle des Semikolons einfach ein Komma.

Beispiel:

```
10    INPUT"Hier die Länge eingeben: ",LAENGE
20    INPUT"Hier die Breite eingeben: ",BREITE
```

Leider gibt es auch Zeichen, die nicht über den Befehl INPUT eingegeben werden können, wie etwa das Komma. Für solche Fälle gibt es weitere Befehle, die im folgenden Abschnitt besprochen werden.

Das nächste Beispiel zeigt, daß mit einem INPUT-Befehl auch mehrere Daten eingegeben werden können. Ein positiver Nebeneffekt ist dabei, daß Sie Speicherplatz sparen.

```
10    INPUT"Bitte 5 Werte eingeben ";A,B,C,D,E
20    PRINT A,B,C,D,E
```

Bis jetzt haben wir uns nur mit kleinen Übungsbeispielen beschäftigt. Da Sie nun jedoch mit der Arbeitsweise und den Anwendungsmöglichkeiten des Befehls INPUT genügend vertraut sind, folgen nun einige Programme, die die praktische Anwendung demonstrieren.

Beispiel 1: Umrechnung von Millimeter in Zoll

Aufgabe:

Schreiben Sie ein Programm, das Sie zuerst nach einer Millimeterangabe fragt. Anschließend soll die Eingabe in die Einheit Zoll umgewandelt werden.
Dabei gilt: 1 Zoll entspricht 25,4 MM

Eingabe:

Millimeter

Verarbeitung:

Die Millimeter-Eingabe in Zoll umwandeln und anschließend beide Einheiten ausgeben.

Ausgabe:

Den eingegebenen Wert in MM
Die umgerechnete Einheit in Zoll

Das Programm:

```
10    CLS
20    INPUT"Wert in MM eingeben: ",MM
30    REM **** Umrechnung in Zoll *****
40    ZOLL = MM / 25,4
50    REM **** Ausgabe der Werte *****
60    PRINT"Eingabe in MM:       ";MM
70    PRINT"Umrechnung in Zoll: ";ZOLL
```

Beschreibung des Programms:

Zeile 10:	Bildschirm löschen
Zeile 20:	Eingabe in Millimeter
Zeile 30:	Kommentarzeile
Zeile 40:	Umrechnung in Zoll
Zeile 50:	Kommentarzeile
Zeile 60:	Ausgabe der eingegebenen Millimeter
Zeile 70:	Ausgabe der Umrechnung in Zoll

Beispiel 2: Einfaches Rechenbeispiel

Das folgende Beispiel soll Ihnen demonstrieren, wie einzelne Angaben später auf dem Bildschirm zusammengesetzt werden und eine Einheit bilden.

Aufgabe:

Schreiben Sie ein Programm, das Ihnen zwei eingegebene Werte multipliziert. Dabei soll das Ergbenis später so auf dem Bildschirm stehen:

```
ZAHL * ZAHL = ERGEBNIS
```

Für die Begriffe Zahl und Ergebnis stehen dann natürlich entsprechende Werte.

Eingabe:

1. Zahl
2. Zahl

Verarbeitung:

Beide Zahlen multiplizieren und das Ergebnis zwischenspeichern

Ausgabe:

Wie oben bereits beschrieben

Das Programm:

```
10    MODE 1
20    CLS
30    INPUT"Bitte erste Zahl eingeben ";ZAHL1
40    INPUT"Und jetzt die zweite Zahl ";ZAHL2
50    REM ******  Jetzt kommt die  **********
60    REM ********* Berechnung **************
70    ERGEBNIS = ZAHL1 * ZAHL2
80    REM *********** Datenausgabe **********
90    PRINT ZAHL1;" * ";ZAHL2;" = ";ERGEBNIS
```

Beschreibung des Programms:

Zeile 10: 40 Zeichen pro Zeile wählen
Zeile 20: Bildschirm löschen
Zeile 30: Erste Zahl eingeben
Zeile 40: Zweite Zahl eingeben
Zeile 50: Kommentarzeile
Zeile 60: Kommentarzeile
Zeile 70: Ergebnis berechnen und speichern
Zeile 90: Ausgabe der kompletten Aufgabe

Bemerkung:

Sehen Sie sich Zeile 90 noch einmal genau an. Hier wird wieder aus mehreren Teilen eine ganze Zeile gebildet. Die Zeichen "*" und "=" sind nur Texte und werden auch als solche behandelt. Als Tren-

nungszeichen dienen hier immer die Semikolons. Die eigentliche Berechnung findet schon in der vorherigen Zeile, der Zeile 70, statt.

Beispiel 3: Gewichtsberechnung

Aufgabe:

Zur Gewichtsberechnung von Stahlprofilen soll ein Profil entwickelt werden. Das Programm soll nach der Länge, Breite und Höhe fragen. Anschließend wird daraus das Gewicht errechnet. Das Material ist Stahl; ein Kubikdezimeter Stahl wiegt 7,85 kg.

Eingabe:

Länge
Breite
Höhe

Verarbeitung:

Erst die Fläche berechnen
Dann das Volumen berechnen
Über das Volumen dann das Gewicht berechnen

Das Programm:

```
10    MODE 1
20    CLS
25    PRINT"Alle Angaben in Dezimeter eingeben"
30    INPUT"Länge des Profils: ",LAENGE
40    INPIT"Breite des Profils:",BREITE
50    INPUT"Höhe des Profils:  ",HOEHE
60    REM **** alle Eingaben gemacht ****
70    REM jetzt die einzelnen Berechnungen
80    FLAECHE=LAENGE*BREITE
90    VOLUMEN=FLAECHE*HOEHE
100   GEWICHT=VOLUMEN*7.85
110   REM *** Datenausgabe komplett *****
120   PRINT"Fläche: ";TAB(20);FLAECHE
130   PRINT"Volumen:";TAB(20);VOLUMEN
140   PRINT"Gewicht:";TAB(20);GEWICHT
```

Beschreibung des Programms:

Zeile 10 bis 20: Bildschirm einstellen
Zeile 25 bis 50: Dateneingabe
Zeile 60 bis 70: Kommentarzeilen
Zeile 80 bis 100: Berechnungen
Zeile 110 bis 140: Datenausgabe

Beispiel 4: Mehrwertsteuer in DM berechnen

Aufgabe:

Schreiben Sie ein Programm, das Ihnen aus einem eingegebenen Bruttobetrag die Mehrwertsteuer (14%) errechnet.

Eingabe:

Bruttobetrag in DM

Verarbeitung:

Erst den Nettobetrag errechnen, dann die Mehrwertsteuer in DM berechnen

Ausgabe:

Bruttobetrag in DM
Nettobetrag in DM
Mehrwertsteuer in DM

Das Programm:

```
10    MODE 1
20    CLS
30    PRINT"Bitte geben Sie den Bruttobetrag"
40    INPUT"in DM ein: ",BRUTTO
50    REM ** jetzt den Nettobetrag errechnen **
60    NETTO = BRUTTO / 114 * 100
70    MWST = BRUTTO - NETTO
80    REM *** Alle Werte ausgeben ***
90    PRINT
100   PRINT
110   PRINT"Bruttobetrag in DM: ";
120   PRINT TAB(30);BRUTTO
130   PRINT"Nettobetrag in DM:  ";
140   PRINT TAB(30);NETTO
150   PRINT"Mehrwertsteuer (14%)";
160   PRINT TAB(30);MWST
```

Beschreibung des Programms:

Zeile 10 bis 20: Bildschirm einstellen
Zeile 30 bis 40: Dateneingabe Brutto
Zeile 50 bis 70: Berechnungen
Zeile 80 bis 160: formatierte Datenausgabe

Bemerkungen:

Beachten Sie die Zeilen 30 und 40. Auch solche Darstellungsformen sind möglich.

5.7 Der Befehl LINE INPUT

Mit diesem Befehl ist es möglich, alle Zeichen der Tastatur einzugeben. LINE INPUT wird jedoch in erster Linie für die Eingabe von Tasten verwendet, in denen z.B. Kommas und andere Zeichen stehen können.

Beispiel:

```
10    MODE 2
20    CLS
30    PRINT"Bitte geben Sie einen beliebigen Text ein: "
40    LINE INPUT TEXT$
50    PRINT
60    PRINT"Sie haben den eben folgenden Text eingegeben:"
70    PRINT CHR$(24);: REM **** REVERSE SCHRIFT EIN *****
80    PRINT TEXT$
90    PRINT CHR$(24):REM **** REVERSE SCHRIFT AUS *****
```

Die Zeilen 70 und 90 bewirken, daß die Darstellung auf Revers umgeschaltet wird. Beim ersten Drucken von CHR$(24) wird diese Position eingeschaltet und beim zweiten Mal wieder aufgehoben. Wie bei INPUT, können auch Texte zur näheren Erläuterung in den Befehl integriert werden.

Beispiel:

```
10    MODE 2
20    CLS
30    LINE INPUT"Bitte Uhrzeit eingeben(HH:MM:SS) : ";ZEIT$
40    PRINT"Eingegebener Text: ";ZEIT$
```

Um das Fragezeichen zu unterdücken, kann auch das Komma verwendet werden.

Beispiel:

```
30 LINE INPUT"Zeit eingeben -->    ",ZEIT$
```

5.8 Der Befehl INKEY$

Auch dies ist ein Eingabe-Befehl; er arbeitet jedoch anders, als wir es von INPUT und LINE INPUT her kennen.

Bei LINE INPUT und INPUT muß jede Eingabe mit der Taste <ENTER> abgeschlossen werden, was bei der Verwendung von INKEY$ nicht notwendig ist. Die Eingabe geschieht mittels eines einzigen Tastendrucks. Das ist allerdings auch der Nachteil dieses Befehls: Sie können nämlich damit nur ein einziges Zeichen oder eine einzige Zahl über die Tastatur eingeben.

Oft wird dieser Befehl benutzt, um im Programm so lange zu warten, bis man sich einen Text auf dem Bildschirm durchgelesen hat. Drückt man dann auf eine beliebige Taste, fährt das Programm mit seiner Ausführung fort.

Im nun folgenden Beispiel sind einige Befehle enthalten, die bisher noch nicht besprochen wurden. Diese Befehle werden jedoch im Anschluß an das Programm genau erläutert.

Beispiel: Abfrage der Tastatur

```
10    PRINT"Diesen Text können Sie sich durchlesen. Wenn Sie"
20    PRINT"fertig sind, drücken Sie bitte eine Taste...."
30    ABFRAGE$=INKEY$
40    IF ABFRAGE$="" THEN GOTO 30
50    PRINT"Ende des Programms."
```

Wenn Sie dieses Programm starten, erscheint zunächst der Text in den Zeilen 10 und 20. In Zeile 30 wird die Tastatur abgefragt. Wird keine Taste gedrückt, geht das Programm wieder zurück in Zeile 30. Bitte beachten Sie, daß zwischen den beiden Anführungsstrichen in Zeile 40 kein Blank (Leerzeichen) stehen darf!

Zeile 30 hätte man auch so schreiben können:

```
30 A$=INKEY$: IF A$="" THEN 30
```

Sie müssen also bei INKEY$ immer mit Textvariablen arbeiten, sonst erhalten Sie eine Fehlermeldung.

5.9 Der Befehl GOTO

Mit dem GOTO-Befehl habe Sie die Möglichkeit, das Programm auf eine bestimmte Zeilennummer umzuleiten. Wichtig ist dabei vor allem, daß diese Zeilennummer auch wirklich vorhanden ist.

Beispiel:

Flächenberechnungen mit Addition

Aufgabe:

Bei der Inventur in einem Materiallager für Bleche sollen die einzelnen Bleche erfaßt und gespeichert werden.

Eingabe:

Länge des Blechs
Breite des Blechs
Anzahl gleicher Bleche

Verarbeitung:

Fläche berechnen für X Stück
Fläche für X Stück laufend aufaddieren
Stückzahl laufend addieren

Ausgabe:

Fläche für X Stück Blech
Laufende Gesamtfläche
Laufende Gesamtstückzahl

Das Programm:

```
10    MODE 1:CLS
20    INPUT"Bitte Länge eingeben:    ",L
30    INPUT"Breite eingeben:         ",B
40    INPUT"Anzahl gleicher Bleche: ";STUECK
50    REM *** Werte berechnen ***
60    FLAECHE = L * B: REM für 1 Stück
70    GFLAECHE = FLAECHE * STUECK
80    GESAMTSTUECK=GESAMTSTUECK + STUECK
90    GESAMTFLAECHE=GESAMTFLAECHE+GFLAECHE
100   PRINT"Fläche gleicher Stücke:";STUECK
110   PRINT"Laufende Gesamtstückzahl:";
120   PRINT GESAMTSTUECK
130   PRINT"Laufende Gesamtfläche";
140   PRINT GESAMTFLAECHE
150   PRINT"------------------------------"
160   GOTO 20
```

Programmbeschreibung:

Mit Zeile 160 zurück (Ende mit zweimal ESC)

5.10 Daten einlesen mit READ und DATA

Oft kommt es beim Programmieren vor, daß beim Austesten von Programmen das Programm wiederholt unterbrochen und geändert werden muß. Anschließend müssen alle Eingaben erneut eingegeben werden, da - wie wir bereits wissen- nach jedem Programm-RUN alle INPUT-Daten automatisch gelöscht werden. Dies kann u.U. sehr zeitraubend sein. Eine komfortablere Methode, fest vorgegebene Daten wiederholt einzulesen, ist mit der READ- und DATA-Anweisung gegeben.

Mit dem Befehl DATA stellt man beliebige Daten zum Lesen bereit. Anschließend kann man diese Daten mit dem Befehl READ wieder lesen. Getrennt werden die Daten in den DATA-Zeilen durch Kommas. Dies ist wichtig, denn wenn ein anderes Zeichen anstelle der Kommas steht, gibt es später Probleme beim Lesen der Daten.

Das folgende Programm stellt drei Zahlen bereit, die gelesen und weiter verarbeitet werden.

```
10    DATA 10, 20, 30
20    READ A
30    READ B
40    READ C
50    REM ******** Verarbeitung der gelesenen Daten ********
60    PRINT A + B + C
70    PRINT A * B - 25 ( A / 2 )
```

Zur Beschreibung des Programms:

Zeile 10 stellt die Daten zum Lesen zur Verfügung. Die Daten werden danach mit dem Befehl READ gelesen. In die Variable A wird der Wert 10 gelesen, in B der Wert 20 und in C der Wert 30.

Der Rechner hat intern einen Zeiger eingebaut, der am Anfang auf die erste Angabe in der DATA-Zeile zeigt. Bei jedem Lesen wandert dieser Zeiger eine Position weiter. Die Bewegung des Zeigers ist jedoch nur daran zu erkennen, daß die aktuelle Ausgabe sich von der vorhergehenden unterscheidet. Die Zeilen 20 bis 40 lesen die Daten in die Variablen ein. In den Zeilen 60 bis 70 wird mit den Daten gerechnet.

Neben rein numerischen lassen sich auch alphanumerische Daten einlesen und verarbeiten.

```
5     MODE 2:CLS
10    DATA München, Berlin, Düsseldorf
20    READ A$
30    READ B$
40    READ C$
50    REM ***** Ausgabe der Daten ******
```

```
60    PRINT A$
70    PRINT B$
80    PRINT C$
```

Das Programm funktioniert genauso wie das vorher beschriebene mit dem Unterschied, daß nun Texte verarbeitet werden.

Auch eine Kombination aus Texten und Zahlen ist möglich, was am folgenden Beispiel demonstriert werden soll.

```
5     MODE 2:CLS
10    DATA Heinrich Zey, Rhedaer Str. 12, 4830, Gütersloh 1
20    DATA Jürgen Noack, Hasenheide 15, 4804, Steinhagen/Amshs.
30    DATA Detlef Buschkamp, Schulstr. 9, 4830, Gütersloh 1
40    REM ******** jetzt die Daten einlesen **********
50    READ N1$,S1$,PLZ1,ORT1$
60    READ N2$,S2$,PLZ2,ORT2$
70    READ N3$,S3$,PLZ3,ORT3$
80    REM ******** alle Daten ausgeben ****************
90    PRINT N1$;" ";S1$;" ";PLZ1;" ";ORT1$
100   PRINT N2$;" ";S2$;" ";PLZ2;" ";ORT2$
110   PRINT N3$;" ";S3$;" ";PLZ3;" ";ORT3$
120   REM ********** Ausgabe beendet *****************
```

In diesem Beispiel werden Texte und Zahlen kombiniert gelesen. Dabei muß darauf geachtet werden, daß der Variablentyp eingehalten wird.

Werden vorhandene Datenbestände erneut gelesen, kommt es oft zu der Fehlermeldung *OUT OF DATA ERROR*. Diese Meldung erscheint, weil der DATA-Zeiger noch auf der letzten Position in der letzten DATA-Zeile steht. Versucht der Rechner dann erneut etwas zu lesen, findet er nichts mehr.

Mit einem Befehl, den wir noch nicht behandelt haben, kann man den DATA-Zeiger wieder auf die erste Position in der ersten DATA-Zeile setzen. Dieser Befehl lautet

RESTORE

Er wird immer dann verwendet, wenn schon einmal gelesene Daten erneut gelesen werden sollen. Zweckmäßigerweise wird der Befehl RESTORE vor dem Lesen eingesetzt.

Beispiel:

```
5     MODE 2: CLS
10    DATA eins, zwei, drei, vier
20    READ A$
30    READ B$
40    READ C$
50    READ D$
60    PRINT A$,B$,C$,D$
```

```
70    REM ******** jetzt den DATA-Zeiger zurücksetzen ******
80    RESTORE
90    READ A$,B$
100   PRINT A$,B$
```

Möchten Sie nur bestimmte und nicht alle Daten noch einmal lesen, können Sie dies mit dem Befehl **RESTORE** und anschließender Angabe der Zeilennummer erreichen. Zum Beispiel würde **RESTORE 200** in einem Programm nur ab Zeile 200 die Daten in den **DATA**-Zeilen lesen.

Ein fehlendes **RESTORE** ist sehr oft Ursache für die Meldung *OUT OF DATA ERROR*. Beachten Sie dies bei Ihren Programmentwicklungen.

5.11 Logische Vergleiche mit IF...THEN

Was wäre ein Computerprogramm wert, wenn es nicht auf Veränderungen von außen reagieren könnte? Stellen Sie sich ein Programm vor, bei dem die Eingaben nicht in irgendeiner Form auf Richtigkeit überprüft würden. Bei Fehleingaben beginnt das reinste Chaos, wenn diese nicht abgefangen werden können. Ein paar Beispiele möchte ich Ihnen kurz schildern, die aus dem Bereich der kaufmännischen und technischen Datenverwaltung stammen und wirklich passiert sind.

Beispiel 1: In einem Programm mußte zur Eingabe der Tagesdaten ein Datum eingegeben werden. Da es sich bei dem Computersystem um ein amerikanisches System handelte, mußte das Datum in der eingedeutschten Programmversion immer in der Form Monat/Tag/Jahr eingegeben werden. Es wäre demgemäß korrekt gewesen, beispielsweise das Datum 12. Juli 1985, in folgender Form einzugeben: 07/12/85.

Da den Sachbearbeitern dieser Umstand aber nicht bekannt war, gaben sie das Datum so ein: 12/07/1985. Erst als bei statistischen Auswertungen schwindelerregende Minusbeträge vom Computer ausgegeben wurden, kam man der Sache auf die Spur. Eine genau Kontrolle des Datums fand in diesem Fall also nicht statt.

Das zweite Beispiel hatte noch schwerwiegendere Folgen; es spielte sich in der Kalkulationsabteilung eines größeren Maschinenbauunternehmens ab. Durch eine fehlerhafte Eingabe in der Vorkalkulation entstand ein erstaunlich günstiger Preis für ein bestimmtes Produkt. Der Kunde vergab sofort seinen Auftrag an das Unternehmen und ließ eine Maschine dort fertigen. Später ergab sich bei der Nachkalkulation, daß der Auftrag von

Anfang an viel zu niedrig angesetzt worden war und das Unternehmen noch Geld zusetzen mußte.

Auch hier war in dem Kalkulationsprogramm eine bestimmte Eingabe nicht überprüft worden. Die Eigenschaft, bestimmte Daten auf Fehleingaben zu überprüfen und entsprechend darauf zu reagieren, macht einen Computer noch leistungsfähiger. Er kann dann entscheiden: Ist eine bestimmte Bedingung erfüllt, dann führe einen bestimmten Befehl aus.

Dazu stellt BASIC zwei Befehle zur Verfügung, die nur zusammen angewendet werden können: `IF` und `THEN`. Ich möchte Ihnen diese Befehle an einem einfachen Beispiel vorstellen.

In einem Programm soll abgefragt werden, ob eine 0 in einer Variablen steht oder nicht. Wird eine 0 festgestellt, soll das Programm mit der Eingabe einer Zahl beginnen. Hier das entsprechende Programm dazu:

```
10    INPUT ZAHL
20    IF ZAHL = 0 THEN GOTO 10
30    PRINT ZAHL
```

Das Programm läßt alle Eingaben, bis auf die Eingabe der 0, bis zum Ende des Programms laufen und wird dort beendet, da nach Zeile 30 nichts mehr erscheint.

Das wichtigste Zeichen ist in diesem Fall das Operationszeichen mit dem Gleichheitszeichen. Weitere Operationscodes werden im folgenden Abschnitt besprochen.

5.11.1 Vergleichsoperatoren

In unserem ersten Beispiel ging es darum, ob zwei Bedingungen übereinstimmen (=). Daneben gibt es noch eine Reihe weiterer, nachfolgend aufgeführter Möglichkeiten:

=	ist gleich
<>	ist ungleich
>	ist größer
<	ist kleiner
>=	ist größer oder gleich
<=	ist kleiner oder gleich

Es ist leicht einzusehen, daß mittels der o.a. Abfragemöglickeiten, je nach Aufgabenstellung, sehr viele verschiedene Zustände überprüft werden können.

Der Befehl IF...THEN wirkt noch mächtiger, wenn er im Zusammenhang mit dem Befehl ELSE eingesetzt wird. ELSE kann nur in Kombination mit IF...THEN verwendet werden. Die Ausführung von IF...THEN...ELSE sieht wie folgt aus:

Wenn (IF) eine Bedingung erfüllt ist, dann (THEN) führe Befehl A aus, andernfalls (ELSE) führe Befehl B aus.

5.11.2 Anwendungsbeispiele mit IF...THEN

In den nun folgenden Beispielen lernen Sie die Anwendung dieser Befehle kennen. Dabei handelt es sich um einfache Programme, die leicht in eigene Entwicklungen übernommen werden können. Im weiteren Verlauf dieses Buches werden Sie sehr oft auf diese drei Befehle stoßen, da sie wirklich sehr leistungsfähig sind.

Beispiel 1: Volumenberechnung

Aufgabe:

Das Volumen eines Raums, das sich aus Länge, Breite und Höhe zusammensetzt, soll berechnet werden. Das Programm soll beendet werden, wenn bei der Länge 0 eingegeben wird.

Eingabe:

Länge, Breite und Höhe des Körpers.

Verarbeitung:

Eingabe der Länge
Abfrage, ob Länge gleich Null
Eingabe der Breite
Eingabe der Höhe
Volumen berechnen
Ergebnis ausgeben
Rücksprung zum Programmanfang

Das Programm:

```
10   MODE 1
```

```
20      INK 0,0
30      BORDER 0
40      CLS
50      INPUT"Bitte geben Sie die Länge ein ",LAENGE
60      REM ist die Länge gleich Null ??
70      IF LAENGE = 0 THEN END
80      INPUT"Bitte jetzt die Breite eingeben ",BREITE
90      INPUT"Wie hoch ist der Raum  ",HOEHE
100     REM  * * * * Volumen berechnen * * * *
110     VOLUMEN=LAENGE*BREITE*HOEHE
120     PRINT VOLUMEN
130     REM * * * * Ausgabe auf Bildschirm * * *
140     GOTO 50
```

Programmbeschreibung:

Die Zeilen 10 bis 40 stellen wieder den Bildschirm und die einzelnen Farben ein. In der folgenden Zeile wird nach der Länge des Raums gefragt. Die Abfrage, die danach kommt, prüft, ob eine Null als Abbruchbedingung eingegeben wurde.

Nachdem die letzten beiden Angaben über INPUT in das Programm eingegeben wurden, kann die Berechnung des Volumens beginnen. Das Ergebnis wird anschließend auf dem Bildschirm ausgegeben. Zum Schluß springt das Programm wieder in Zeile 50 und beginnt erneut mit der Arbeit.

Beispiel 2: Vorwärts zählen

Aufgabe:

Ein Programm soll entwickelt werden, das von 1 bis 100 zählt.

Eingabe:

Entfällt, da das Programm dies besorgt

Verarbeitung:

Zählerstand um eins erhöhen
Zähler auf Bildschirm ausgeben
Abfrage, ob schon bei 100 angekommen
Rücksprung zum Zähler

Ausgabe:

Aktuellen Zählerstand ausgeben

Das Programm:

```
10      MODE 1
20      CLS
```

```
30    INK 0,0
40    BORDER 0
50    ZAEHLER = ZAEHLER + 1
60    PRINT ZAEHLER
70    IF ZAEHLER = 100 THEN END
110   GOTO 50
```

Programmbeschreibung:

In Zeile 50 wird der aktuelle Zählerstand um eins erhöht. Dieser Stand wird ausgegeben und anschließend auf seine Endbedingung abgefragt. Bei 100 wird das Programm beendet. Ist der Stand noch unter 100, wird mit Zeile 50 forgefahren.

Beispiel 3: Rückwärts zählen

Aufgabe:

Entwickeln Sie ein Programm, das von 20 bis 0 mit einer Schrittweite von 0.5 rückwärts zählt.

Eingabe:

Entfällt hier und wird vom Programm übernommen.

Verarbeitung:

Zähler um 0.5 verringern
Zähler auf dem Bildschirm ausgeben
Abfrage, ob Zähler schon bei 0 ist
Wenn nicht, Zähler weiter verringern

Ausgabe:

Laufenden Zählerstand

Das Programm:

```
10    MODE 1
20    CLS
30    BORDER 0
40    INK 0,0
50    ZAEHLER=20
60    ZAEHLER=ZAEHLER-0.5
70    PRINT ZAEHLER
80    IF ZAEHLER = 0 THEN END
90    GOTO 60
```

Programmbeschreibung:

Das Programm arbeitet wie Beispiel 2, mit dem Unterschied, daß es hier subtrahiert. Dafür ist Zeile 60 zuständig. Beachten Sie, daß in Zeile 50 der Anfangswert festliegen muß.

Beispiel 4: Abfragen mit IF...THEN...ELSE

Aufgabe:

In den letzten Beispielen haben Sie gesehen, wie man mit `IF...THEN` Bedingungen abfragt. Dabei wurde immer nach dem Motto gehandelt, wenn... dann.
Im folgenden Beispiel geht es um eine erweiterte Möglichkeit: wenn... dann... sonst. Übersetzt heißt eine solche Abfrage `IF...THEN...ELSE`.
Wir wollen uns nun ein Programm ansehen, in dem eine Wiederholung vorkommt. Dabei soll zunächst die Mehrwertsteuer aus einem Rechnungsbetrag errechnet werden. Nach der Ausgabe der Daten soll das Programm abfragen, ob eine weitere Berechnung durchgeführt werden soll.

Eingabe:

Bruttobetrag

Verarbeitung:

Netto=Brutto / 114 * 100
MWST = Brutto - Netto

Ausgabe:

Mehrwertsteuer

Das Programm:

```
10    MODE 1:CLS
20    INPUT"Bruttobetrag in DM ";BRUTTO
30    REM *** Berechnung ***
40    NETTO=BRUTTO / 114 * 100
50    MWST=BRUTTO-NETTO
60    PRINT"Mehrwertsteuer: ";MWST
70    PRINT:PRINT"Nochmal ? j/n"
80    E$=INKEY: IF E$="" THEN 80
90    IF E$="n" THEN END ELSE GOTO 10
```

Beschreibung:

Zeile 90: Wenn n eingegeben wird, dann das Programm beenden, sonst wieder in Zeile 10 zurückgehen und neu beginnen.

5.12 Programmieren von Schleifen

In den vorherigen Kapiteln haben wir schon die Programmierung von Wiederholungen mit der Abfrage IF...THEN kennengelernt. Zur Erinnerung noch einmal ein Beispiel, in dem bis 10 gezählt wird.

```
10    N=0
20    N=N+1
30    PRINT N
40    IF N=10 THEN END
50    GOTO 20
```

Der Zähler in Zeile 20 addiert bei jedem Durchlauf eine 1 zu seinem vorherigen Inhalt, so daß die Summe immer größer wird. In Zeile 30 wird der aktuelle Wert des Zählers auf dem Bildschirm ausgegeben.

Zeile 40 prüft nach, ob die Endbedingung schon eingetreten ist. Wenn ja, wird das Programm mit dem Befehl END beendet. Liegt dagegen der Wert des Zählers unter 10, dann erfolgt der Sprung zurück in Zeile 20, in der wieder eine 1 zum aktuellen Wert hinzuaddiert wird.

Eine weitere Art der Programmierung möchte ich Ihnen in diesem Abschnitt vorstellen. Es geht dabei um sogenannte FOR...NEXT-Schleifen, die nachfolgend behandelt werden sollen.

Löschen Sie Ihr eben eingegebenes Programm mit NEW und geben Sie stattdessen das ein:

```
10    FOR N=1 TO 10
20    PRINT N
30    NEXT N
```

Starten Sie nun das Programm mit RUN und sehen Sie sich auf dem Bildschirm das Ergebnis an. Auch dieses Programm zählt bis 10, nur wesentlich eleganter.

Eine Schleife setzt sich aus folgenden Angaben zusammen:

1. dem Befehl FOR
2. der *Laufvariablen*
3. einem *Gleichheitszeichen*
4. einem *Anfangswert*
5. dem Befehl TO
6. einem *Endwert*
7. dem Befehl STEP
8. der Angabe der *Schrittweite*

Die Punkte 7 und 8 sind nicht unbedingt erforderlich und können auch weggelassen werden. Ist dies der Fall, wird für die Schrittweite vom Computer standardmäßig der Wert 1 gewählt.

Die Punkte 1 bis 8 sind jedoch nur der erste Teil einer Schleife. Man nennt dies das Öffnen einer Schleife. Da aber eine geöffnete Schleife auch wieder geschlossen werden muß, verwendet man den Befehl NEXT, dem die Angabe der Variablen folgt. In unserer Aufstellung fehlen also noch die Punkte 9 und 10.

9. Mit dem Befehl NEXT die Schleife schließen.
10. Mit der Angabe der Variablen mitteilen, welche Schleife geschlossen werden soll.

Noch ein paar Bemerkungen zu den Laufvariablen: Die geläufigste Laufvariable ist I. Für den ungeübten Programmierer ist es aber hilfreicher, wenn er aussagekräftige Variablen benutzt, die später zur besseren Lesbarkeit eines Programms beitragen. Hier einige Beispiele für aussagekräftige Variablen:

```
a) FOR DURCHLAUF = 1 TO 100

b) FOR PAUSE = 1 TO 2000

c) FOR ABSTAND = 1 TO 4
```

Sie können hier sofort erkennen, was diese Schleife bewirkt.

Eine Schleife benötigt einen festen Anfangs- und einen Endwert. Ob diese Werte nun in Form von Variablen oder Konstanten vorhanden sind, ist im Prinzip egal. Auf jeden Fall müssen sie dem Rechner bekannt sein, sonst kommt es zu Fehlern im Programm.

Bei jedem Durchlauf der Schleife prüft der Rechner intern, ob der Endwert bereits erreicht wurde. Diesen Vorgang könnte man mit der Abfrage `IF...THEN` vergleichen, nur ist er auf dem Bildschirm nicht sichtbar. Ist der Endwert noch nicht erreicht, erhöht sich automatisch die Laufvariable um 1, falls nichts anderes vereinbart wurde.

Wurde der Endwert erreicht, wird mit dem Befehl NEXT mit Angabe der entsprechenden Variablen die Schleife wieder geschlossen. Achtung: Schleifen müssen immer mit NEXT geschlossen werden!

Schleifen können überall dort eingesetzt werden, wo mehrere gleiche Vorgänge zu bearbeiten sind. Dies spart unter Umständen erheblichen Speicherplatz im Programm.

Auf den nächsten Seiten finden Sie eine Reihe von Programmbeispielen, die mit diesen Schleifen arbeiten. Zum besseren Verständnis beginnen wir mit einigen recht einfachen Übungen.

5.12.1 Zählen mit einer Schleife bis 500

Aufgabe:

Schreiben Sie ein kurzes Programm, mit dem von 1 bis 500 gezählt werden soll. Jeder aktuelle Schleifenwert soll auf dem Bildschirm ausgegeben werden.

Eingabe:

Entfällt, weil vom Programm übernommen

Verarbeitung:

Mit FOR...NEXT und LOCATE arbeiten

Ausgabe:

Es soll der jeweils aktuelle Wert der Schleife auf dem Bildschirm erscheinen.

Das Programm:

```
10    FOR ANZAHL = 1 TO 500
20    LOCATE 10,10
30    PRINT ANZAHL
40    NEXT ANZAHL
```

Programmbeschreibung:

Zeile 10:	Schleife für 500 Durchläufe öffnen
Zeile 20:	Textcursor auf dem Bildschirm positionieren
Zeile 30:	Aktuellen Wert der Schleife auf dieser Position ausgeben
Zeile 40:	Schleife wieder schließen

5.12.2 Zählen mit einer bestimmten Schrittweite

Aufgabe:

Schreiben Sie ein Programm, das von 0 nach 50 mit einer Schrittweite von 2 zählt.

Eingabe:

Entfällt hier, da diese Aufgabe vom Programm übernommen wird

Verarbeitung:

Mit der Schleife FOR...NEXT und dem Zusatz STEP arbeiten

Ausgabe:

Alle Schleifenwerte mit geringem Abstand

Das Programm:

```
10     MODE 2
20     CLS
30     FOR ZAHL = 0 TO 50 STEP 2
40     PRINT ZAHL,
50     NEXT ZAHL
```

Beschreibung des Programms:

Zeile 10 und 20: Bildschirm einstellen
Zeile 30: Schleife öffnen
Zeile 40: Aktuellen Wert ausgeben
Zeile 50: Schleife schließen

5.12.3 Rückwärts zählen mit einer Schleife

Aufgabe:

Es soll ein Programm entwickelt werden, das später von 1000 nach 500 mit einer Schrittweite von -10 zurück zählt.

Eingabe:

Entfällt hier, da vom Programm erzeugt

Verarbeitung:

Schleife mit dem Zusatz STEP verwenden

Ausgabe:

Alle durchlaufenden Schleifenwerte

Das Programm:

```
10     MODE 2
20     FOR ZAEHLEN = 1000 TO 500 STEP -10
30     PRINT ZAEHLEN,
40     NEXT ZAEHLEN
```

Beschreibung des Programms:

Zeile 10: Auf 80 Zeichen umschalten
Zeile 20: Schleife öffnen
Zeile 30: Ausgabe des aktuellen Wertes
Zeile 40: Schleife wieder schließen

Hinweise:

Beachten Sie, daß bei einer negativen Schrittweite (STEP-...) der Anfangswert größer sein muß, als der Endwert! Wird dies nicht beachtet, kommt es zu Fehlern.

5.12.4 Programmieren einer Schleife mit variablen Ausgaben

Aufgabe:

Statt der bisher fest vorgegebenen Angaben bei einer Schleife sollen im folgenden Beispiel alle Angaben, d.h. Anfangs- und Endwert sowie die Schrittweite, frei sein.

Eingabe:

Anfangswert - Endwert - Schrittweite

Verarbeitung:

Über INPUT alle geforderten Daten eingeben. Mit diesen Werten dann in die Schleife gehen.

Ausgabe:

Alle aktuellen Schleifenwerte

Das Programm:

```
10    MODE 2
20    INPUT"Bitte Anfangswert eingeben ";ANFANG
30    INPUT"Endwert eingeben ";ENDWERT
40    INPUT"Schrittweite eingeben ";SCHRITT
50    REM **** Schleife öffnen ****
60    FOR I = ANFANG TO ENDWERT STEP SCHRITT
70    PRINT I,
80    NEXT I
90    REM ***** Schleife schließen *****
```

Beschreibung des Programms:

Zeile 10: Auf 80 Zeichen schalten
Zeile 20: Anfangswert eingeben
Zeile 30: Endwert eingeben

Zeile 40: Schrittweite eingeben
Zeile 50: Kommentarzeile
Zeile 60: Schleife öffnen
Zeile 70: Aktuellen Wert ausgeben
Zeile 80: Schleife wieder schließen
Zeile 90: Kommentarzeile ausgeben

Hinweise:

Sie können alle Variablennamen durchspielen, die denkbar sind. Auch negative Schrittweiten können eingegeben werden, wenn der Anfangswert größer als der Endwert ist.
Wir haben in den ersten beiden Beispielen bereits die Arbeitsweise einer Schleife kennengelernt. Damit wir den Vorgang besser erkennen können, wurde jedesmal der aktuelle Schleifenwert ausgegeben.
In den nun folgenden Beispielen sollen die Schleifen etwas praxisbezogener sein.

5.12.5 Quadratflächen berechnen mit einer Schleife

Aufgabe:

Von den Quadraten mit der Seitenlänge von 10 bis 20 sollen die jeweiligen Flächen berechnet werden. Die Einheit spielt in diesem Beispiel keine Rolle.

Eingabe:

Entfällt, weil die Schleife diese Aufgabe übernimmt

Verarbeitung:

Berechnung nach der Formel für die Fläche F = A * A

Ausgabe:

Seitenlänge und Fläche für jedes Quadrat

Das Programm:

```
10    MODE 2
20    FOR A = 10 TO 20
30    F = A * A
40    PRINT"Seitenlänge ";A;TAB(30);"Fläche ";F
50    PRINT STRING*(60,"=")
60    NEXT A
```

Beschreibung des Programms:

Zeile 10: 80 Zeichen wählen
Zeile 20: Schleife öffnen für den Bereich von 10 bis 20
Zeile 30: Fläche berechnen aus den aktuellen Werten der Seitenlänge
Zeile 40: Ausgabe der Seitenlänge und der berechneten Fläche
Zeile 50: Linie aus 60 Zeichen ausgeben
Zeile 60: Schleife wieder schließen

Hinweise:

Zeile 40 können Sie auch so schrieben:

```
40    PRINT"Das Quadrat mit der Seitenlänge";
41    PRINT A;"hat die Fläche von ";F
```

5.12.6 Summe aller Zahlen im Bereich von 0 bis 50 berechnen

Aufgabe:

Es sollen alle Zahlen von 0 bis 50 aufaddiert werden.
Beispiel: 0+1+2+3+4+5+.......+50

Eingabe:

Wird von der Schleife erledigt

Verarbeitung:

Siehe Beispiel oben

Ausgabe:

Laufende Summe und Gesamtsumme

Das Programm:

```
10    MODE 2
20    FOR ZAHL = 0 TO 50
30    SUMME=SUMME+ZAHL
40    LOCATE 1,5
50    PRINT"Aktuelle Zahl: ";ZAHL
60    LOCATE 1,8
70    PRINT"Lfd. Summe  : ";SUMME
80    NEXT ZAHL
```

Beschreibung des Programms:

Zeile 10: 80 Zeichen wählen
Zeile 20: Schleife für 50 Durchläufe öffnen
Zeile 30: Zur letzten Summe die aktuelle Zahl addieren

Zeile 40: Textcursor positionieren
Zeile 50: Aktuelle Zahl ausgeben
Zeile 60: Textcursor positionieren
Zeile 70: Laufende Summe ausgeben
Zeile 80: Schleife schließen

Hinweise:

Die Zeile 75 verzögert den Ablauf des Programms:

```
75    FOR T = 1 TO 1000: NEXT T
```

Diese Zeile kann wahlweise eingegeben werden.

5.12.7 Programmieren einer Warteschleife

Im letzten Beispiel hatten Sie die Möglichkeit, Zeile 75 einzugeben. Diese Zeile stellt eine Verzögerung dar, weil innerhalb dieser Schleife nichts geschieht.

Die Zeit, die das Programm zum Zählen braucht, wird hier als Pause für den Bildschirm benutzt. Hat die Schleife schließlich den Endwert erreicht, geht es im Programm weiter. Warteschleifen sind deshalb sehr nützliche Einrichtungen, weil man sich in aller Ruhe die Informationen auf dem Bildschirm ansehen kann.

Beim Schneider CPC 464, der recht schnell arbeitet, empfehle ich Ihnen Warteschleifen, die bis 3000 zählen. Dieser Wert ist für die meisten Anwendungen ausreichend.

Sie sollten auch wieder eine aussagekräftige Laufvariable wählen, damit Sie später besser erkennen können, was in der jeweiligen Zeile gerade geschieht.

Hier sind nun drei Beispiele für Warteschleifen:

```
a) FOR PAUSE = 1 TO 3000: NEXT PAUSE
b) FOR WARTEN = 1 TO 4000: NEXT WARTEN
c) FOR VERZOG = 1 TO 3000: NEXT VERZOG
```

Nachteilig an den Warteschleifen ist folgendes: Ist die Warteschleife einmal durchlaufen worden, geht es sofort im Programm weiter, auch wenn Sie Ihren Text auf dem Bildschirm noch nicht zuende gelesen haben. Dem

kann man nur begegnen, wenn man die Endwerte der Schleife etwas höher ansetzt.

5.12.8 Berechnen des Mittelwertes verschiedener Eingaben

Aufgabe:

Sie sollen ein Programm schreiben, das von einer vorher festgelegten Anzahl von Werten den Mittelwert errechnet.

Eingabe:

Angabe, aus wievielen Zahlen der Mittelwert berechnet werden soll.

Verarbeitung:

Mit einer Schleife die Anzahl der Werte eingeben und anschließend die Gesamtsumme aller Zahlen durch die Anzahl der gemachten Eingaben teilen.

Ausgabe:

Anzahl der Eingaben
Gesamtsumme
Mittelwert

Das Programm:

```
10    MODE 2
20    INPUT"Wieviele Werte eingeben ";ANZAHL
30    FOR I=1 TO ANZAHL
40    CLS
50    PRINT"Bitte den";I;".ten Wert eingeben:"
60    PRINT
70    INPUT"Wert --->  ";WERT
80    SUMME=SUMME+WERT
90    NEXT I
100   CLS
110   PRINT"A U S W E R T U N G"
120   PRINT"==================="
130   PRINT
140   PRINT"Sie haben";
140   PRINT ANZAHL;"Werte eingegeben"
160   PRINT
170   PRINT"Das ergibt die Summe von ";
180   PRINT SUMME;" und einen Mittelwert von ";
190   PRINT SUMME / ANZAHL
```

Beschreibung des Programms:

Zeile 10: 80 Zeichen wählen

Zeile 20: Eingabe der Anzahl Werte
Zeile 30: Schleife öffnen
Zeile 40: Bildschirm löschen
Zeile 50: Textausgabe
Zeile 60: Leerzeile auf dem Bildschirm
Zeile 70: Wert eingeben
Zeile 80: Wert zur Summe addieren
Zeile 90: Schleife schließen
Zeile 100: Bildschirm löschen
Zeile 110: Textausgabe
Zeile 120: Unterstreichen der Ausgabe
Zeile 130: Leerzeile auf dem Bildschirm
Zeile 140: Textausgabe
Zeile 150: Anzahl der Werte ausgeben
Zeile 160: Leerzeile auf dem Bildschirm
Zeile 170: Textausgabe
Zeile 180: Summe ausgeben
Zeile 190: Mittelwert berechnen und ausgeben

5.12.9 Einlesen von festen Daten

Aufgabe:

Es sollen eine Reihe von Daten eingelesen werden, die in den DATA-Zeilen abgelegt sind.

Eingabe:

Entfällt

Verarbeitung:

Mit einer Schleife und dem Befehl READ alle Daten lesen, die als DATA-Werte angegeben sind.

Ausgabe:

Alle eingegebenen Werte

Das Programm:

```
10    MODE 2
20    DATA Werner, Karl-Heinz, Rolf, Ulrike
30    DATA Karin, Heike, Thomas, Eberhard
40    FOR LESEN = 1 TO 8
50    READ NAME$
60    PRINT LESEN;". ter Name: ";NAME$
70    NEXT LESEN
```

Beschreibung des Programms:

Zeile 10: 80 Zeichen wählen
Zeile 20: DATA-Werte
Zeile 30: DATA-Werte
Zeile 40: Schleife zum Lesen öffnen
Zeile 50: Name aus den DATA-Zeilen lesen
Zeile 60: Namen und dessen Position ausgeben
Zeile 70: Schleife wieder schließen

Hinweise:

Diese Möglichkeit der "Eingabe" von Daten bietet sich besonders bei Testläufen an, bei denen man nicht immer die gleichen Daten erneut eingeben möchte.

5.12.10 Verschachtelte Schleifen

Die normale Schleife ist uns jetzt bekannt. Es gibt jedoch auch die Möglichkeit, mehrere Schleifen miteinander zu kombinieren. Dazu ein kleines Programmbeispiel:

```
10    MODE 2
20    FOR AUSSEN = 1 TO 2      <----------------------*
30         FOR INNEN = 1 TO 5     <---------*
40         PRINT"*";
50         NEXT INNEN             <---------*
60    PRINT
70    NEXT AUSSEN              <----------------------*
```

Dieses Programm arbeitet mit verschachtelten Schleifen. Wichtig dabei ist die Reihenfolge, in der die Schleifen geöffnet und vor allem wieder geschlossen werden. Komplizierte Berechnungen lassen sich oft mit verschachtelten Schleifen auf ein Minimum begrenzen.

Beachten Sie bitte folgendes:

Die zuletzt genannte Schleife muß zuerst wieder geschlossen werden, sonst kommt es zu Fehlern im Ablauf. Außerdem dürfen Laufvariablen bei verschachtelten Schleifen nicht doppelt vergeben werden.

Ich empfehle Ihnen, bei verschachtelten Schleifen nach den Zeilennummern ein wenig Platz zu lassen, damit die Struktur der Verschachtelung besser erkennbar ist.

5.12.11 Kleines Einmaleins in Tabellenform

Aufgabe:

Das kleine Einmaleins soll in Tabellenform auf dem Bildschirm ausgegeben werden.

Eingabe:

Nichts, wird durch das Programm vorgegeben

Verarbeitung:

Zwei Schleifen werden miteinander kombiniert, d.h. die aktuellen Werte der Schleife werden multipliziert.
Zusätzlich wird der aktuelle Wert einer Schleife für die Funktion TAB benutzt.

Ausgabe:

In Tabellenform

Das Programm:

```
10    MODE 2
20    CLS
30    INK 0,0
40    BORDER 0
50    FOR I=1 TO 10
60    PRINT TAB(6*I);I;
70    NEXT I
80    PRINT TAB(4);STRING$(59,"*")
90    FOR I=1 TO 10
100   FOR Z=1 TO 10
110   PRINT TAB(Z*6);I*Z;
120   NEXT Z
130   PRINT
140   NEXT I
150   LOCATE 1,3
160   FOR I=1 TO 10
170   PRINT I;"**"
180   NEXT I
```

5.13 Der Befehl END

Oft ist es notwendig, ein Programm irgendwo zu beenden. Dazu steht Ihnen der Befehl END zur Verfügung.

Beispiel:

```
10    N=N+1
20    PRINT N
30    IF N=10 THEN END
40    GOTO 10
```

Wenn in diesem Programmbeispiel die Variable N den Wert von 10 angenommen hat, wird automatisch das Programm mit END beendet. Bei Verwendung von END erscheint keine Meldung auf dem Bildschirm, sondern nur einfach READY.

5.14 Der Befehl ERASE

Innerhalb von größeren Programmen haben Sie mit ERASE die Möglichkeit, nicht mehr benötigte Variablen wieder zu löschen und für neue Aufgaben zur Verfügung zu stellen. Das folgende Beispiel soll Ihnen kurz die Wirkungsweise vor Augen führen:

```
10    A=10
20    PRINT"Alter Wert von A ist:  ";A
30    ERASE A
40    PRINT"Neuer Wert von A:      ";A
```

Das Programm löscht in Zeile 30 den Inhalt der Variablen A. Es gibt den Inhalt dann in Zeile 40 aus, damit sie sich davon überzeugen können, daß die Variable wirklich gelöscht wurde.

5.15 Die Schleifenanweisung WHILE....WEND

Ähnlich wie bei einer FOR....NEXT-Schleife wird mit dieser Schleifenanweisung eine Wiederholung programmiert.

Während die FOR....NEXT-Schleife einen bestimmten Anfangs- und Endwert besitzt, arbeiten die Schleifen mit WHILE und WEND etwas anders.

Hier wird die Schleife erst verlassen, wenn eine bestimmte Bedingung zutrifft. Diese Bedingung wird in der Anweisung WHILE formuliert, die die Schleife öffnet. Ähnlich wie bei den bereits besprochenen FOR....NEXT-Schleifen muß auch eine mit WHILE geöffnete Schleife wieder geschlossen werden. Man erreicht dies mit der Anweisung WEND.

Beispiel: Tastaturabfrage

```
10      CLS
20      PRINT "START...."
30      WHILE INKEY$=""
40      WEND
50      PRINT"ENDE...."
```

Die Zeile 30 öffnet hier die Schleife. Wenn keine Taste gedrückt wird, ergibt sich eine Endlosschleife, weil die Bedingung in dieser Zeile nicht zutraf. Die Zeile 40 schließt die mit WHILE geöffnete Schleife wieder.

Beispiel: Bestimmte Zufallszahl

```
10      CLS
20      WHILE R <> 50
30      R=INT(RND(1)*100)
40      PRINT R,
50      WEND
60      PRINT"ENDE"
```

Dieses Programm erzeugt so lange eine Reihe von Zufallszahlen, bis die in der Zeile 20 angegebene Bedingung nicht mehr stimmt.

5.16 Der Befehl TIME

Mit dem Einschalten des Systems fängt auch eine eingebaute Uhr an zu laufen. Sie hat beim Einschalten den Wert Null und zählt nun in Schritten von 1/300 Sekunde die Zeiteinheiten, die bisher vergangen sind.

Mit Hilfe dieses Befehls kann man sich so z.B. die Zeit anzeigen lassen, die seit dem Einschalten vergangen ist. Man erreicht dies, indem man einfach vor den Befehl TIME einen PRINT-Befehl setzt.

Beispiel: PRINT TIME

Da aber nun die Zeit noch nicht in der gewünschten Form angegeben wird, teilen wir alles noch einmal durch 300, damit auf dem Bildschirm auch die Zeit in Sekunden erscheint.

Beispiel: Zeitverbrauch ermitteln für eine Schleife

```
10      ANFANGSZEIT=TIME
20      PRINT"Programmstart...."
30      FOR I=1 TO 3000:NEXT I
40      ENDZEIT=TIME
```

```
50    VERBRAUCH=ENDZEIT-ANFANGSZEIT
60    VERBRAUCH=VERBRAUCH / 300
70    PRINT"Verbrauchte Zeit in Sekunden: ";VERBRAUCH
```

In der Zeile 10 wird der Beginn des Zählvorgangs zeitlich festgehalten und einer Variablen zugewiesen.

Die Zeile 30 ist für die Bearbeitungszeit zuständig. Wenn die Schleife abgearbeitet worden ist, dann wird in der folgenden Zeile erneut die aktuelle Zeit festgehalten.

Die Differenz aus Endzeit und Anfangszeit ergibt dann die verbrauchte Zeit, die dann nur noch in Sekunden umgewandelt werden muß.

Anmerkung: Die Zeit für das Laden und Speichern von Programmen wird nicht mitgezählt. Bitte beachten Sie das bei Ihren Programmentwicklungen.

5.17 Der Befehl CHR$

Jedes Zeichen hat im Computer eine bestimmte Nummer. Diese Nummer nennt man ASCII-Code. ASCII bedeutet American Standard Code for Information Interchange. Viele Computer arbeiten mit diesem Standard, wenn nicht ausschließlich, so doch teilweise. Meistens kann man davon ausgehen, daß die Zeichen ab dem ASCII-Code 32 bis zum ASCII-Code 126 gleich sind. Ausnahmen bestätigen auch hier wieder die Regel. Im Bereich 32 bis 126 liegen die Satzzeichen, die Zahlen, das große und kleine Alphabet, mathematische Zeichen und einige andere Zeichen. Sie werden nun vielleicht fragen, welche Bedeutung die ASCII-Zeichen im Bereich über 126 und unter 32 haben. Beginnen wir mit letzterem. Diesen Bereich benutzen alle Systeme für sogenannte Steuerfunktionen, mit denen man beispielsweise einen Piepton erzeugen oder den Cursor an den Anfang der aktuellen Zeile setzen kann. So hat etwa der ASCII-Code 24 beim Schneider die Aufgabe, den Modus REVERS ein- und auszuschalten. Bei einem anderen System, dem Commodore 64, wird der ASCII-Code 18 benutzt, um die REVERS-Funktion aufzurufen.

Die Versionen der einzelnen Hersteller unterscheiden sich also im Bereich von 0 bis 31. Die größten Überraschungen bringt jedoch der Bereich 127 bis 255. Nebenbei bemerkt, der maximale Wert liegt bei 255. Der ASCII-Code besteht also aus insgesamt 256 verschiedenen Zeichen und Funktionen, da das System bei der Zählung immer bei Null beginnt.

Doch zurück zu unserem Bereich von 127 bis 255. In diesem Bereich belegen die Hersteller der einzelnen Computer den Zeichensatz mit den außergewöhnlichsten Zeichen. Da gibt es neben dem griechischen Alphabet noch Bomben, grinsende Gesichter, Musiknoten, tanzende Strichmännchen, und nicht zu vergessen, ein wahres Heer von Grafikzeichen in jeder Ausführung. Was mich persönlich am Schneider stört, sind die fehlenden Zeichen, die es beispielsweise ermöglichen, eine Zahl ins Quadrat zu erheben. Man muß in solchen Fällen immer schreiben "11 QM" oder sich ein Zeichen selbst umdefinieren. Auf Teile des griechischen Alphabets oder auf eines der tanzenden Strichmännchen könnte ich dagegen verzichten. Aber dies ist vermutlich Ansichtssache, denn für mich zählt in ersten Linie der praktische Einsatz eines Computers. Der eine oder andere Leser, der vielleicht Spiele programmieren möchte, ist jedoch bestimmt recht dankbar für eine fertige Bombe oder ein kleines Männchen.

Immerhin steht Ihnen aber mit diesem Zeichensatz ab dem ASCII-Code 127 eine sehr breite Palette von Sonder- und Grafikzeichen zur Verfügung. Alle diese Zeichen kann man mit dem Befehl `CHR$()` darstellen, wobei in der Klammer die Nummer des entsprechenden ASCII-Codes steht. Wenn Sie z.B. den Befehl PRINT `CHR$(65)` im Direktmodus eingeben und <ENTER> drücken, erscheint als Ergebnis der Buchstabe "A" auf dem Bildschirm. Der `CHR$`-Befehl hat folgendes Darstellungsformat:

```
CHR$(Nummer)
```

Das anschließende Programm gibt Ihnen alle Zeichen ab dem ASCII-Code 32 auf dem Bildschirm aus.

```
10 MODE 1
20 INK 0,0
30 BORDER 0
40 CLS
50 FOR CODE=32 TO 255
60 PRINT"ASCII-CODE : ";CODE;" ergibt ---> ";CHR$(CODE)
70 FOR PAUSE=1 TO 1000
80 NEXT PAUSE
90 NEXT CODE
```

Ein großer Nachteil aller Computersysteme ist, daß sich die Grafikzeichen ab dem ASCII-Code 127 nicht immer alle ausdrucken lassen. Technische Schwierigkeiten entstehen dann, wenn z.B. nicht der Schneider, sondern ein fremder Drucker benutzt wird, der genau auf dieser Stelle in seinem Zeichensatz ein anderes Zeichen hat. Die meisten Probleme gibt es bei der Anpassung von verschiedenen Druckern, die später einmal von ein und demselben Programm angesteuert werden sollen. Oft erlebt man dann

haarsträubende Geschichten, zumal es manchmal sogar Unterschiede beim gleichen Druckertyp geben kann (Baujahr, Version).

5.18 Die Befehle LEFT$, RIGHT$ und MID$

Um Texte beliebig bearbeiten zu können, stehen bei fast allen Computersystemen im BASIC einige nützliche Befehle zur Verfügung.

So ist es zum Beispiel für ein Sortierprogramm kein Problem, nur die ersten vier oder fünf Buchstaben eines Nachnamens beim Sortieren zu beachten. Ein anderes Programm überschreibt innerhalb einer Textzeile einen bestimmten Begriff mit einem ebenso langen Ersatzbegriff, oder es zerlegt ein Wort in seine einzelnen Buchstaben und setzt daraus dann neue Begriffe zusammen.

Alle angesprochenen Beispiele lassen sich mit den Befehlen LEFT$, RIGHT$ und MID$ realisieren. Die einzelnen Befehle haben dabei folgende Aufgaben:

LEFT$	schneidet von einer Zeichenkette von links eine bestimmte Anzahl von Zeichen ab.
RIGHT$	arbeitet wie LEFT$, nur von rechts aus.
MID$	nimmt aus einem Text einen beliebigen Teil heraus.

Wir wollen nun auf den folgenden Seiten gemeinsam ein Programm entwickeln, daß alle drei Befehle am praktischen Beispiel demonstriert. Unser Ausgangsbegriff lautet:

AUTOKINOPARKPLATZWAECHTERWITWE

Geben Sie dazu das folgende Programm ein:

```
10 MODE 1
20 INK 0,0
30 BORDER 0
40 CLS
50 A$="AUTOKINOPARKPLATZWAECHTERWITWE"
60 PRINT A$
```

Sie erhalten als Ergebnis dieses Programms den Inhalt der Zeile 50 auf dem Bildschirm. Als nächstes wollen wir versuchen, aus unserem Begriff das Wort AUTO herauszulösen. Mit dem BASIC-Befehl LEFT$ ist dies kein Problem, denn wir haben damit die Möglichkeit, aus einem Text einen Teiltext abzuschneiden. Das Format für den Befehl LEFT$ sieht so aus:

LEFT$(Textvariable, Anzahl Zeichen von links aus gerechnet)

Für unser Beispiel bedeutet das:

```
50 A$="AUTOKINOPARKPLATZWAECHTERWITWE"
60 PRINT LEFT$(A$,4)
```

Nach dem Programmstart sollte jetzt das Wort AUTO auf Ihrem Bildschirm erscheinen. Aus dem Ursprungstext in Zeile 50 wurde also eine Zeichenkette von vier Zeichen Länge abgeschnitten. Der Befehl LEFT$ sagt dem Rechner, daß er von vorne abschneiden soll.

Im zweiten Schritt soll das Wort WITWE aus dem großen Text herausgelöst werden. Da das Wort WITWE am Ende des Ursprungstextes steht, kann nicht mit LEFT$ gearbeitet werden. Der Befehl, der jetzt benutzt werden muß, lautet RIGHT$; er hat das gleiche Format wie LEFT$. RIGHT$ schneidet einen Teiltext jedoch von rechts ab. Unsere nächste Programmzeile lautet daher:

```
70 PRINT RIGHT$(A$,5)
```

Wenn Sie das Programm erneut starten, erscheint als Ergebnis der Zeile 70 das Wort WITWE auf dem Bildschirm.

RIGHT$ arbeitet also im Prinzip wie der Befehl LEFT$, nur mit dem Unterschied, daß neue Teiltexte von rechts aus entstehen. Zusätzlich ist es möglich, die Teiltexte in einer Textvariablen zu speichern und sie so für die weitere Bearbeitungen zur Verfügung zu stellen.

Unsere beiden neuen Begriffe AUTO und WITWE sollen einmal in zwei verschiedenen Variablen gespeichert werden. Die dazu nötigen Programmzeilen sehen dann so aus:

```
80 B$=LEFT$(A$,4)
90 C$=RIGHT$(A$,5)
```

Nun stehen die beiden Begriffe zur weiteren Verarbeitung jederzeit zur Verfügung. Sie können beispielsweise aneinander gesetzt werden und damit ein neues Wort bilden. Versuchen Sie es einmal:

```
100 PRINT B$+C$
```

Wird das ganze Programm erneut gestartet, erscheint neben den schon bekannten Begriffen nun das Wort AUTOWITWE.

Wir haben also gesehen, mit welchen Befehlen es möglich ist, Teiltexte links und rechts von einer vorhandenen Zeichenkette abzuschneiden.

Möchte man aber eine beliebige Zeichenfolge mitten aus einem vorhandenen Text herauslösen, benötigt man dazu einen weiteren Befehl.

Dieser Befehl greift sich von einer bestimmten Position an eine bestimmte Anzahl an Zeichen heraus. Damit ist man wesentlich flexibler als mit den beiden anderen Befehlen. Unser neuer Befehl heißt MID$ und hat folgendes Darstellungsformat:

```
MID$(Textvariable, 1. Position des neuen
Teiltextes, Anzahl Zeichen ab dieser
Position)
```

Dazu ein Beispiel: Aus der Zeichenkette in Zeile 50 möchten Sie das Wort PARK herausnehmen. Für die Verwendung des Befehls MID$ brauchen Sie hierfür noch folgende Angaben:

a) Aus welcher Variablen soll der Teiltext herausgenommen werden?

b) Ab der wievielten Position sollen die Zeichen herausgenommen werden?

c) Wieviele Zeichen sollen ab der Position, die unter b) steht, herausgenommen werden?

Im Klartext bedeutet das für unser Programm, daß wir aus der Zeichenkette in Zeile 50 ab der 9. Position innerhalb der Zeichenkette 4 Zeichen herausnehmen möchten. Die dafür notwendige Programmzeile lautet:

```
110 PRINT MID$(A$,9,4)
```

Da das Wort PARK ab der 9. Position beginnt und insgesamt 4 Zeichen lang ist, wurde Zeile 110 so geschrieben, wie Sie sie oben sehen.

Falls gewünscht, könnte man diesen Teiltext ebenfalls wieder in einer Variablen zur späteren Verarbeitung ablegen. Auch dies wollen wir einmal versuchen. Dabei könnte man Zeile 120 beispielsweise so schreiben:

```
120 E$=MID$(A$,9,4)
```

Sie sehen, wie nützlich eine solche Funktion ist, mit der man aus einem bereits vorhandenen Text beliebig viele Teiltexte bilden kann.

Als nächsten Schritt wollen wir noch verschiedene Wörter aus der Ursprungszeichenkette in Zeile 50 bilden. Dabei habe ich an folgende Wörter gedacht:

1. PLATZ --- ablegen in der Variablen D$
2. KINO --- ablegen in der Variablen G$

3. WAECHTER --- ablegen in der Variablen F$

Umgesetzt in die Programmiersprache BASIC sehen unsere nächsten Programmzeilen so aus:

```
130 D$=MID$(A$,13,5):REM PLATZ
140 G$=MID$(A$,5,4): REM KINO
150 F$=MID$(A$,18,8):REM WAECHTER
160 PRINT D$
170 PRINT G$
180 PRINT F$
```

Nach dem Programmstart mit RUN müßten die letzten Begriffe, die auf dem Bildschirm stehen, lauten: PLATZ, KINO und WAECHTER.

Sollten diese drei Wörter nicht richtig erscheinen, haben Sie sich vermutlich in den Zeilennummern 130 bis 150 irgendwo verschrieben. Meistens geschieht dies bei der Angabe der Position oder bei der Länge des Teiltextes.

Alle Begriffe, die wir im Verlauf des Programms in einer Variablen zwischengespeichert haben, können wir jetzt miteinander kombinieren. Wir wollen im folgenden Schritt drei neue Begriffe auf dem Bildschirm ausgeben. In den Zeilennummern 190 bis 210 wird dies praktisch realisiert.

```
190 PRINT E$+D$:REM PARK und PLATZ
200 PRINT G$+D$:REM KINO und PLATZ
210 PRINT D$+F$:REM PLATZ und WAECHTER
```

Texte lassen sich zusammensetzen, indem man das Zeichen "+" zwischen die einzelnen Variablen setzt. Es wird hier natürlich nicht mit einem Text gerechnet, sondern der Text wird lediglich aneinandergesetzt. Undenkbar sind beispielsweise folgende Schreibweisen:

PRINT D$-E$ oder PRINT A$*F$ oder PRINT D$/A$

Beachten Sie das bitte bei der Zusammensetzung einzelner Variablen! Auch Variablen mit einem Semikolon können zusammengesetzt werden. Falls Ihnen diesenicht mehr so geläufig sind, sehen Sie sich nochmals das Kapitel an, in dem der Befehl PRINT erklärt wird. Dort finden Sie Hinweise für die Verwendung des Semikolons.

Bisher haben wir immer ganze Begriffe aus einem vorhandenen Text herausgelöst. Es ist aber auch möglich, nur bestimmte Buchstaben daraus zu isolieren. Auch hier kommt die Funktion MID$ wieder zur Anwendung, mit dem Unterschied, daß man bei der Angabe der Zeichenlänge einfach dem Programm eine 1 mitteilt. Dadurch können vollkommen neue

Begriffe entstehen, die mit unserem Text in Zeile 50 fast nichts mehr gemeinsam haben.

Wenn wir einmal den Begriff "ZENTRALEINHEIT" entstehen lassen wollen, benötigen wir dazu von folgenden Buchstaben die folgende Menge:

Anzahl	**Buchstaben**
1	Z
3	E
2	N
2	T
1	R
1	A
1	L
2	I
1	H

Um unser erstes Zeichen, den Buchstaben Z, auf dem Bildschirm auszugeben müßten wir `PRINT MID$(A$,17,1)` eingeben. Mit diesem Befehl teilen wir dem Rechner mit, daß er ab der 17. Position aus der Variablen A$ ein einziges Zeichen, in unserem Fall das Z, herausnehmen soll.

Unsere Zeilennummern, die jetzt folgen, greifen sich aus dem Text in der Variablen A$ jeweils an einer bestimmten Position ein Zeichen heraus und drucken es auf dem Bildschirm aus. Zur Verhinderung des Zeilenvorschubs am Bildschirm, steht nach jedem Ende einer Programmzeile nach dem `PRINT` ein Semikolon, dessen Bedeutung Sie ja schon kennen. Geben Sie nun die folgenden Programmzeilen in den Rechner ein:

```
220 PRINT
230 PRINT"Der Begriff  - ZENTRALEINHEIT -  wird aus dem"
240 PRINT"Begriff  - AUTOKINOPARKPLATZWAECHTERWITWE - "
250 PRINT"neu gebildet:  "
260 PRINT
270 PRINT MID$(A$,17,1);:REM Buchstabe Z
280 PRINT MID$(A$,20,1);:REM  Buchstabe E
290 PRINT MID$(A$,7,1);:REM Buchstabe N
300 PRINT MID$(A$,3,1);:REM Buchstabe T
310 PRINT MID$(A$,11,1);:REM Buchstabe R
320 PRINT MID$(A$,1,1);:REM Buchstabe A
330 PRINT MID$(A$,14,1);:REM Buchstabe L
340 PRINT MID$(A$,20,1);:REM Buchstabe E
350 PRINT MID$(A$,6,1);:REM Buchstabe I
360 PRINT MID$(A$,7,1);:REM Buchstabe N
370 PRINT MID$(A$,22,1);:REM Buchstabe H
380 PRINT MID$(A$,20,1);:REM Buchstabe E
390 PRINT MID$(A$,6,1);:REM Buchstabe I
```

```
400 PRINT MID$(A$,3,1);:REM Buchstabe T
```

Natürlich hätte man auch einzelne Befehle in einer Zeile hintereinander schreiben können. Der Übersicht halber wurde aber darauf verzichtet.

Mit dem Befehl MID$ können Sie ebenfalls Teile im Ursprungstext verändern. Diese Möglichkeit ist nicht so bekannt, wie beispielsweise das Herausnehmen von Textstücken. Um Ihnen diesen Vorgang an einem Beispiel zu demonstrieren, löschen Sie bitte die Zeilennummern 60 bis 400 mit dem DELETE-Befehl. Mit DELETE 60-400 wird dieser Bereich der Zeilennummern gelöscht und steht nun wieder zur Verfügung. Damit wir unsere Ursprungszeichenkette auch noch zu sehen bekommen, geben Sie die Zeilennummern 60 und 70 wie folgend ein:

```
60 PRINT A$
70 PRINT
```

Zeile 70 dient dazu, optisch eine Leerzeile auf dem Bildschirm auszugeben.

Die Aufgabe, die ich Ihnen jetzt stellen möchte, lautet: Ersetzen Sie in der Zeichenkette "AUTOKINOPARKPLATZWAECHTERWITWE" das Wort KINO durch einen ebenso langen, beliebigen anderen Begriff. Den neuen Begriff geben Sie vorher mit INPUT ein.

Eine erste Lösungsmöglichkeit wäre, mit LEFT$, RIGHT$ und dem neuen Begriff das Wort neu zusammenzusetzen. Dies möchte ich Ihnen nachfolgend kurz vorstellen, obwohl es nicht die beste Lösung ist. Sie können aber daran erkennen, daß es oft mehrere Wege gibt, die zum gewünschten Erfolg führen. Es ist sicherlich recht nützlich, diesen Weg einmal nachzuvollziehen, da er zum besseren Verständnis der zweiten Variation führt. Hier nun das Programm für den ersten Lösungsweg:

```
60    PRINT A$
70    PRINT
80    B$=LEFT$(A$,4)
90    C$=RIGHT$(A$,22)
100   PRINT B$
110   PRINT C$
120   PRINT"-----------------------------------"
130   PRINT
140   INPUT"Bitte geben Sie nun den Ersatzbegriff ein: ",begriff$
150   A$=B$+begriff$+C$
160   PRINT"Der Ursprungstext lautet nun so:"
170   PRINT A$
```

Erläuterung zur Arbeitsweise des Programms:

In Zeile 80 und 90 werden die ersten vier Zeichen von vorne und die zweiundzwanzig Zeichen von hinten abgeschnitten. Es handelt sich dabei

um diejenigen Zeichen, die sich nicht ändern. Zur Kontrolle werden diese beiden Variablen noch ausgedruckt. In Zeile 140 wird nach dem neuen Begriff gefragt, den man an dieser Stelle später sehen möchte. Neben Zeile 140 ist Zeile 150 die wichtigste, da hier der neue Begriff wieder aus einzelnen Stücken zusammengesetzt wird. Aus den Elementen AUTO und PARKPLATZWAECHTERWITWE zuzüglich des neuen Begriffs entsteht dann ein neues Wort, welches der Variablen A$ zugewiesen wird.

In Zeile 170 erscheint das Ergebnis Ihrer Arbeit.

Die Lösung ist zwar nicht die eleganteste, aber sicherlich die für Sie am einfachsten überschaubare. Es folgt nun der zweite, professionellere Weg. Löschen Sie dafür zunächst die Zeilennummern 80 bis 130. Das Programm sieht dann so aus:

```
80    INPUT"Bitte geben Sie den Ersatzbegriff ein :  ",begriff$
90    MID$(A$,5,4)=begriff$
100   PRINT A$
```

Wenn Sie dieses Programm mit RUN starten und einen vierstelligen neuen Begriff eingeben, wird in der Variablen A$ genau an der angegebenen Position der Ersatzbegriff eingefügt. Im Prinzip kann man also in einem solchen Fall von "Überschreiben" sprechen, da das Wort KINO vom neuen Begriff überschrieben wird.

Ein weiteres Beispiel für die Verwendung von MID$ ist der Einsatz für ein Programm zur Erzeugung von Laufschrift. Unter Laufschrift versteht man einen Text, der Zeichen für Zeichen auf dem Bildschirm erscheint und damit den Eindruck erweckt, er laufe über den Bildschirm. Das Programm lautet wie folgt:

```
10    MODE 1
20    INK 0,0
30    BORDER 0
40    CLS
50    INPUT"Bitte geben Sie den Text für die Laufschrift ein : ",text$
60    FOR L=1 TO LEN(TEXT$)
70    PRINT MID$(TEXT$,L,1);
80    FOR PAUSE=1 TO 70
90    NEXT PAUSE
100   NEXT L
110   PRINT
120   PRINT"========================================"
130   GOTO 50
```

Programmbeschreibung zum Laufschriftprogramm:

Die Zeilen 10 bis 40 schalten nur den Bildschirmmodus ein und färben den Rahmen und den Hintergrund schwarz. Dazu wird der Bildschirm gelöscht.

Zeile 50 ist Ihnen schon aus unserem letzten Programm bekannt. Hier wird der Austauschbegriff eingegeben.

Neu ist für Sie der Schluß von Zeile 60. Hier wird mit einer Schleife gearbeitet, die zunächst die Länge der eingegebenen Zeichen ausrechnet. Dies ist nötig, um festzustellen, wie oft die Schleife durchlaufen werden soll.

Mit dem Befehl LEN(TEXTVARIABLE) stellt der Computer fest, aus wievielen Zeichen die untersuchte Zeichenkette besteht. Dabei werden auch Leerstellen als Zeichen behandelt. Mehr zu diesem Befehl auf den nächsten Seiten.

Und nun die wichtigste Zeile des ganzen Programms: Zeile 70. Hier wird mit Hilfe des Befehls MID$ jeweils ein Zeichen aus dem eingegebenen Text, der nun in der Variablen TEXT$ steht, auf den Bildschirm gebracht. Dabei spielt die Laufvariable L eine große Rolle, denn sie teilt dem Computer mit, ab der wievielten Position ein Zeichen aus dem eingegebenen Text herausgenommen werden soll. Bei jedem Schleifendurchlauf erhöht sich der Wert von L um 1, was dazu führt, daß das nächste Zeichen auf dem Bildschirm erscheint.

Die Zeilennummern 80 und 90 stellen eine Warteschleife dar, damit man sehen kann, daß der Text auch wirklich Zeichen für Zeichen wiedergegeben wird. Die Geschwindigkeit wird durch den Endwert in Zeilen 80 bestimmt. Sie können diesen Wert versuchshalber einmal erhöhen und sich dann von der Wirkung überzeugen.

Zeile 100 schließt die äußere Schleife. Zeile 110 und 120 erzeugen einen Leerraum auf dem Bildschirm, wenn die Laufschrift fertig ist.

Zeile 130 teilt dem Schneider mit, daß er wieder in Zeile 50 springen soll, in der nach dem nächsten Text gefragt wird. Falls Sie nichts mehr eingeben möchten, drücken Sie zweimal die Taste <ESC>.

5.19 Der Befehl INSTR

Dieser Befehl fehlt in vielen BASIC-Versionen anderer Hersteller, obwohl er für die Bearbeitung von Zeichenketten m.E. einer der angenehmsten Befehle überhaupt ist. INSTR sucht innerhalb einer Variablen alle Zeichen, die mit dem Suchzeichen identisch sind und gibt die Position an, an der diese Zeichen im Text stehen. Mögliches Einsatzgebiet: In einer Textverarbeitung sollen alle Textstellen, die mit "ss" in der aktuellen Zeile stehen, durch den Buchstaben "ß" ersetzt werden.

Es folgt ein Beispiel, in dem bestimmte Zeichen oder Begriffe innerhalb unserer schon bekannten Zeile 50 gesucht werden sollen.

Das Programm:

```
10    MODE 1
20    BORDER 9
30    INK 0,0
40    CLS
50    A$="AUTOKINOPARKPLATZWAECHTERWITWE"
60    PRINT"Welches Wort oder welches Zeichen"
70    INPUT"suchen Sie:   ",SUCH$
80    SUCH$=UPPER$(SUCH$)
90    STELLE=INSTR(A$,SUCH$)
100   IF STELLE=0 THEN GOTO 160
110   PRINT
120   PRINT"Der Suchbegriff beginnt in dem Text"
130   PRINT"an der";STELLE;"-ten Stelle."
140   PRINT
150   GOTO 60
160   PRINT"Diesen Buchstaben oder Begriff gibt"
170   PRINT"es leider im Text nicht."
180   PRINT
190   GOTO 60
```

Programmbeschreibung:

In Zeile 70 wird der Suchbegriff eingegeben, der in der folgenden Zeile noch in Großbuchstaben umgewandelt wird. Dies ist nötig, weil der Text in der Variablen A$ auch in Großbuchstaben steht und es sonst Komplikationen während des Vergleichs gibt. Die Position, an der der Suchbegriff im Text steht, wird einer Variablen zugewiesen, die in unserem Fall STELLE heißt. Wenn es keine Stelle gibt, an der dieser Buchstabe oder diese Zeichenfolge auftritt, bleibt die Variable auf Null.

In Zeile 100 wird abgefragt, ob der Wert der Variablen gleich Null ist. Wenn ja, dann befindet sich kein Buchstabe oder kein Text dieses Suchbegriffs im Text von Zeile 50. Das Programm geht in diesem Fall in Zeile

160, wo eine entsprechende Nachricht für den Anwender ausgegeben wird, und beginnt dann wieder von vorn.

Hat das Programm ein Zeichen oder eine Zeichenfolge gefunden, dann wird die Stelle angegeben, an der die gefundene Zeichenfolge im Text beginnt. Anschließend kehrt das Programm wieder in Zeile 60 zurück und fragt nach dem neuen Suchbegriff.

Nochmals zur Erinnerung die Zeile 90:

```
INSTR(Variable, in der gesucht werden soll,
Suchbegriff)
```

Alles in allem gesehen, ist dies eine angenehme Arbeitserleichterung für die Zeichenkettenbe- und -verarbeitung.

5.20 Der Befehl STRING$

In vielen Programmen werden immer wieder Linien benötigt, die über den ganzen Bildschirm gehen. Die gängigste Lösung wäre eine Schleife, die zum Beispiel 40mal ein beliebiges Zeichen druckt. Ein entsprechendes Programm könnte so aussehen:

```
10    FOR I=1 TO 40
20    PRINT"-";
30    NEXT I
40    PRINT"ENDE"
```

Das Programm zeichnet eine Linie, die aus 40 Minuszeichen besteht. Der Nachteil dieser Lösung besteht darin, daß der nächste PRINT-Ausdruck sofort hinter dem letzten Minuszeichen erscheint. Im Modus 1 fiel Ihnen dies vielleicht nicht so auf, da das Wort ENDE ja in der nächsten Zeile am Anfang stand. Wenn Sie jedoch einmal mit MODE 2 auf die 80-Zeichen-Darstellung umschalten und das Programm erneut starten, sehen Sie die Bescherung. Abhilfe schafft hier nur ein PRINT zwischen dem NEXT und der nächsten PRINT-Anweisung, in unserem Fall in Zeile 35: 35 PRINT. Nun spielt es keine Rolle mehr, wieviele Zeichen Sie ausgeben und in welchem Bildschirmmodus Sie sich gerade befinden.

Im Prinzip ist dies ein recht umständliches Verfahren, wie Sie soeben gesehen haben. Da der Schneider aber einen komfortablen BASIC-Befehlssatz hat, gibt es auch für dieses kleine Problem einen schönen Befehl. Mit diesem Befehl kann man eine Linie aus bestimmten Zeichen darstellen; er hat folgendes allgemeines Format:

`STRING$(Anzahl der Zeichen, gewünschtes Zeichen)`

Für die Anzahl der Zeichen können Sie maximal den Wert 255 wählen. Nach dem Komma folgt das gewünschte Zeichen, wobei es eine Reihe unterschiedlicher Angabemöglichkeiten gibt, von denen ich Ihnen einige vorstellen möchte.

Mit dem Befehl `PRINT STRING$(40,"-")` wird eine Linie gezogen, die aus Minuszeichen besteht und 40 Zeichen lang ist. Eine weitere Möglichkeit besteht darin, die Nummer eines Zeichens aus dem Zeichensatz anzugeben. Mit dem Befehl `PRINT STRING$(20,77)` erhalten Sie 20 Buchstaben (M) auf dem Bildschirm. Und zuletzt noch einige fest eingebaute Grafikzeichen: `PRINT STRING$(40,95)` oder `PRINT STRING$(40,131)` oder `PRINT STRING$(40,159)`

Damit Sie sich von der Vielzahl der Variationen überzeugen können, habe ich das folgende kleine Programm entwickelt. Es zeigt Ihnen alle darstellbaren Zeichen des Zeichensatzes von Nummer 32 bis Nummer 255 und zwar jeweils 25 Zeichen einer Nummer. Zusätzlich wird die aktuelle Nummer angezeigt, die gerade durchlaufen wird. Dieses Programm können Sie beispielsweise benutzen, um sich für Linien die besten Darstellungen auszusuchen.

```
10    MODE 1
20    INK 0,0
30    BORDER 0
40    CLS
50    FOR I=32 TO 255
60    PRINT I;"    ";STRING$(25,i)
70    PRINT
80    FOR PAUSE=1 TO 1000:NEXT PAUSE
90    NEXT I
```

Kernstück des Programms ist Zeile 60, in der die Nummer des aktuellen Zeichens und eine Zeichenkette von 25 Zeichen ausgegeben wird. Damit die Ausgabe lesbar bleibt, wurde die Warteschleife in Zeile 80 installiert.

Zusammenfassend kann man sagen, daß der `STRING$`-Befehl eine wirkliche Erleichterung ist, die zudem noch schneller arbeitet als eine Schleife, die eine Linie erzeugt. Nutzen Sie deshalb diese Erleichterung, wann immer es möglich ist.

5.21 Der Befehl ASC

Mit diesem Befehl können Sie den ASCII-Code eines Zeichens ermitteln. Der Befehl hat folgendes Darstellungsformat:

```
ASC(Zeichenkette)
```

Beispiel:

```
PRINT ASC("X")
```

Sie erhalten als Ergebnis den ASCII-Code für das Zeichen X. Zur Überprüfung können Sie in Ihrem Handbuch nachschlagen und das Ergebnis des Rechners mit der dort aufgeführten Tabelle vergleichen.

Ein anderes Beispiel:

```
10    MODE 1
20    INK 0,0
30    BORDER 0
40    CLS
50    DATA A,B,C,D,E,F,G,H,I,J
60    FOR LESEN=1 TO 10
70    READ C$
80    PRINT ASC(C$)
90    NEXT LESEN
```

Nach dem Programmstart erhalten Sie die Zahlen 65 bis 74. Die Werte entsprechen dem ASCII-Code dieser Zeichen. Sehen Sie sich dazu auch wieder die Tabelle im Schneider-Handbuch an, und kontrollieren Sie das Ergebnis.

Das oben abgebildete Programm liest nacheinander die Zeichen A bis J ein und ermittelt mit der Funktion **ASC** deren ASCII-Wert. Experimentieren Sie ein wenig mit dem Programm und verändern Sie Zeile 50. Geben Sie dabei aber mindestens 10 andere Zeichen ein, da bei weniger Daten die Fehlermeldung *"DATA exhausted in 70"* erscheint.

Es folgt ein weiterer Anwendungsfall, bei dem die Funktion dieses Befehls am praktischen Beispiel verdeutlicht wird.

Aufgabe:

Bei einer Eingabe sollen nur die Buchstaben J und N zulässig sein. Mit **ASC** ist die Prüfung vorzunehmen.

Verarbeitung:

Eingabe J oder N
Ist das erste Zeichen gleich dem ASCII-Code für J?
Ist das erste Zeichen gleich dem ASCII-Code für N?
Wenn nicht, dann neue Eingabe verlangen

Ausgabe:

Welches Zeichen wurde gedrückt? J oder N

Das Programm:

```
10   MODE 1
20   INK 0,0
30   BORDER 0
40   CLS
50   INPUT"Bitte antworten Sie:   J = ja    N     = nein ",E$
60   IF ASC(E$)=74 THEN GOTO 110
70   IF ASC(E$)=78 THEN GOTO 120
80   PRINT"Leider falsch. Nur J oder N eingeben."
90   PRINT
100  GOTO 50
110  PRINT"Der Buchstabe J wurde gedrückt.":END
120  PRINT"Der Buchstabe N wurde gedrückt."
```

Programmbeschreibung:

Aus der Eingabe in Zeile 50 wird der ASCII-Wert gebildet. Ist er gleich 74, verzweigt das Programm in Zeile 110; ist er gleich 78, verzweigt es in Zeile 120 und druckt hier die passende Meldung aus.

Trifft keine der beiden Bedingungen zu, dann wird eine Fehlermeldung für den Benutzer ausgegeben und nach einer neuen Eingabe gefragt. Der Befehl END in Zeile 110 verhindert die Ausführung von Zeile 120, die sonst als nachfolgende Zeile abgearbeitet würde.

Auf eine Besonderheit möchte ich Sie in diesem Zusammenhang noch hinweisen. Es ist auch möglich, mit kompletten Zeichenketten zu arbeiten, von denen aber jedesmal nur das erste Zeichen der Zeichenkette zur Ermittlung des ASCII-Codes herangezogen wird.

Beispiel:

```
10 INPUT"Bitte geben Sie ein Wort ein    ";WORT$
20 PRINT ASC(WORT$)
```

Sie sehen, nur das erste Zeichen wird beachtet!

5.22 Der Befehl STR$

Manchmal ist es erforderlich, Zahlen in Texte einzubinden. Dabei ist es unvorteilhaft, numerische und alphanumerische Daten miteinander zu verbinden, wie etwa PRINT A$+Z+B$.

Beispielsweise ist für eine Textverarbeitung ein rein alphanumerischer Text gefragt, in dem die oben beschriebenen Kombinationen nicht vorkommen sollen. Stellen Sie sich einen Handwerker vor, der mit seinem Schneider Angebote schreiben möchte. Dazu muß er die Möglichkeit haben, Zahlen in seine Textverarbeitung einzubauen, von denen er später vielleicht noch die Gesamtsumme bilden möchte. Am einfachsten wäre es, er würde seine Werte als Text eingeben und alle Berechnungen im Kopf oder per Taschenrechner ausrechnen. Im Zeitalter der Computer wäre dies jedoch Zeitverschwendung, denn wozu kann der Computer rechnen?

Obige Aufgabe wollen wir in einem der nächsten Beispiele dem Computer überlassen, denn er soll ja der Arbeitserleichterung dienen und keine zusätzliche Arbeit verursachen.

Um ein solches Beispiel aufführen zu können, müssen wir zunächst wissen, wie der Befehl STR$ funktioniert; er hat folgendes Format:

```
STR$(Numerischer Wert)
```

oder

```
STR$(Numerische Variable)
```

Wichtig ist dabei, daß innerhalb der Klammer ein numerischer Wert vorkommen muß. Ob dieser Wert direkt dort steht oder als Variable dargestellt wird, ist nicht von Bedeutung. STR$ wandelt den rein numerischen Wert in einen alphanumerischen Wert um. Das bedeutet, daß beispielsweise aus einer numerischen 10 eine Folge von zwei Zeichen, nämlich 1 und 0, entsteht.

Zum besseren Verständnis dazu ein Beispiel:

```
10    MODE 1
20    INK 0,0
30    BORDER 0
40    CLS
50    INPUT"Bitte geben Sie eine Zahl ein     ";ZAHL
60    PRINT"Sie haben den Wert ";ZAHL;" eingegeben."
70    ZEICHEN$=STR$(ZAHL)
80    PRINT ZEICHEN$
```

Dieses kleine Programm fragt in Zeile 50 nach einer Zahl, die als numerischer Wert eingegeben wird. Zur Kontrolle wird Ihnen die Zahl noch einmal in der folgenden Zeile ausgedruckt. In Zeile 70 wandelt das Programm die numerische Eingabe aus Zeile 50 in einen alphanumerischen Wert um und weist die neue Zeichenfolge der Variablen ZEICHEN$ zu. Am Ende des Programms wird zum Ausdruck der Variablen ZEICHEN$ aufgefordert. Auf dem Bildschirm erhalten Sie genau den Wert, den Sie in Zeile 50 über INPUT als numerischen Wert eingegeben haben, nur handelt es sich jetzt um eine Folge von Zeichen, also um einen völlig anderen Variablentyp als zuvor bei der Eingabe.

Mit der Variablen ZEICHEN$ ist natürlich die Einbindung in eine Textverarbeitung wesentlich einfacher, als mit einem numerischen Wert. Jetzt kann man jederzeit diese alphanumerische Variable für weitere Verarbeitungen nutzen.

Unser Beispiel können wir noch ausbauen. Im nächsten Schritt werden die beiden Variablen ZAHL und ZEICHEN$ weiter verarbeitet.

```
90    PRINT ZEICHEN$;" + ";ZEICHEN$;" ergibt hier ";ZEICHEN$+ZEICHEN$
100   PRINT ZAHL;" + ";ZAHL;" ergibt hier ";ZAHL+ZAHL
```

Wenn Sie das Programm gestartet und einen Wert eingegeben haben, sehen Sie bereits den Unterschied. In Zeile 90 werden zwei Texte zusammengefügt, in denen der Eingabewert steht. In Zeile 100 aber wird richtig addiert, denn hier handelt es sich um numerische Daten, bei denen bekanntlich mathematische Operationen keine Problemem darstellen.

Und nun zurück zu unserem Handwerker - sagen wir einem Malermeister. Er möchte Angebote schreiben, in denen er die Streich- oder Tapezierfläche in Quadratmetern und den Quadratmeterpreis für diese Tätigkeit angibt. Später soll das Angebot als Ausdruck auf dem Bildschirm erscheinen und zwar als reiner Text ohne numerische Angaben. Alle Zahlen müssen demgemäß in einen Text umgewandelt werden.

Dieses kleine Programm läßt sich noch weiter ausbauen. Man könnte das erstellte Angebot auch als Text auf Diskette oder Kassette abspeichern und somit für wiederholte Wiederverwendung sichern. Außerdem wäre natürlich der Punkt Druckerausgabe eine interessante Aufgabe, die man sich vornehmen könnte. Hier sollten Sie sich selbst einmal einige Gedanken machen, was das Programm noch alles erledigen könnte. Nachfolgend nun das Listing für unseren Malermeister:

```
10    MODE 2
20    INK 0,0
30    BORDER 0
40    CLS
```

```
50    INPUT"Wandflaeche in QM :     ",FL
60    INPUT"Preis pro QM            ",QP
70    SUMME=INT(FL*QP)*100+0.5)/100
80    TEXT1$=STR$(FL)
90    TEXT2$=STR$(QP)
100   TEXT3$=STR$(SUMME)
110   ZEILE$(0)=CHR$(24)+"Arbeitsaufwand:"+CHR$(24)
120   ZEILE$(1)="Wandflaechen vorbereiten, kleine Putzschaeden mit Fuellstoff
      ausbessern."
130   ZEILE$(2)="Flaeche mit normaler Rauhfasertapete fachgerecht auf Stoß
      tapezieren."
140   ZEILE$(3)="Alle Preise incl. Tapetenlieferung"
150   ZEILE$(4)=STRING$(70,32)
160   ZEILE$(5)="Quadratmeter: "+TEXT1$+STRING$(7," ")+"Preis pro QM:"
      +TEXT2$+STRING$(12,32)+"Summe: "+TEXT3$+" DM"
170   BR=SUMME*1.14
180   BR=INT(BR*100+0.5)/100
190   ZEILE$(6)=STRING$(79,154)
200   ZEILE$(7)="Die Gesamtsumme betraegt incl. 14% MWST beträgt"+STR$(BR)
      +" DM"
210   ZEILE$(8)=STRING$(79,"=")
220   CLS
230   FOR ZEILE=0 TO 8
240   PRINT ZEILE$(ZEILE)
250   NEXT ZEILE
```

Wenn Sie das Programm eingegeben haben, starten Sie es mit RUN und geben für die Wandfläche und den Quadratmeterpreis die gewünschten Daten ein. Das Programm errechnet jetzt automatisch die Gesamtsumme.

Programmbeschreibung:

Die Zeilen 10 bis 40 sind Ihnen bereits bekannt. In den folgenden Zeilen wird nach der Wandfläche und nach dem Quadratmeterpreis gefragt. Zeile 70 errechnet den Betrag, der sich ergibt, wenn die Fläche mit dem Preis pro Quadratmeter multipliziert wird. Dieser Betrag wird auf zwei Stellen nach dem Komma gerundet, wie es bei DM-Beträgen üblich ist.

In den Zeilen 80, 90 und 100 werden die eingegebenen numerischen Werte mit Hilfe des Befehls `STR$` in einen Text umgewandelt.

Zeile 110 weist einer Textvariablen den Text "Arbeitsaufwand" zu, wobei dieser später revers auf dem Bildschirm ausgegeben wird, weil die beiden Angaben `CHR$(24)` vor und hinter dem Text stehen. Mit dem ersten `CHR$(24)` wird umgeschaltet in den REVERS-Modus, mit dem zweiten wird wieder in die normale Darstellung zurückgeschaltet.

In den Zeilen 120 bis 140 wird ebenfalls einer Textvariablen ein Text zugewiesen. Zeile 150 erzeugt eine Folge von 70 Leerzeichen, die in einer Variablen gespeichert werden, wie alle anderen Zeichen auch.

Zeile 160 ist das Kernstück dieses Programms. Hier werden verschiedene Teiltexte zu einer ganzen Textzeile zusammengesetzt. Diese Zeile besteht im einzelnen aus dem Wort "Quadratmeter: ", der umgewandelten Fläche, einer Folge von 7 Leerzeichen, dem Wort "Preis pro QM: ", dem umgewandelten Betrag für einen Quadratmeter, einer Folge von 12 Leerzeichen, dem Wort "Summe: ", dem umgewandelten Summenwert sowie den Zeichen " DM" am Schluß der Zeile.

Die Zeilen 170 und 180 errechnen den Bruttobetrag und runden diesen auf 2 Stellen nach dem Komma, wie bei dem Nettobetrag. Zeile 190 erzeugt eine Folge von 79 Grafikzeichen vom ASCII-Code 154. Zeile 200 setzt sich wieder zusammen aus einem Text und dem umgewandelten Betrag in DM. Zeile 210 erzeugt 79 Zeichen des Gleichheitszeichens.

In den Zeilen 220 bis 250 werden alle Textzeilen ausgegeben.

5.23 Der Befehl VAL

Wir wissen nun, wie eine Zahl in einen Text umgewandelt wird. Nachfolgend möchte ich Ihnen nun die umgekehrte Funktion zeigen. Der Befehl VAL wandelt eine Zahl, die als Textvariable vorliegt, wieder in einen numerischen Wert zurück. Er hat folgendes Format:

```
VAL(Textvariable)
```

oder

```
VAL(Zeichenkette)
```

Probieren Sie die folgenden Befehle einmal im Direktmodus, also ohne eine Zeilennummer davor, aus.

```
PRINT VAL("23.25")

PRINT VAL("20")+VAL("20")

A$="16": PRINT VAL(A$)*2
```

Wie Sie sicherlich bemerkt haben, wird die Zeichenkette in einen numerischen Wert zurückverwandelt. Mit dem numerischen Wert können anschließend alle mathematischen Berechnungen durchgeführt werden. Ein Beispiel dazu:

```
10    MODE 1
20    INK 0,0
```

```
30    BORDER 0
40    CLS
50    INPUT"Bitte geben Sie einen Wert ein:   ",Z$
60    PRINT"Der eben eingegebene Wert wird als Text behandelt."
70    PRINT"Addiert ergibt dies ";Z$+Z$
80    PRINT
90    PRINT"Nun wird die Zeichenkette in einen numerischen
100   PRINT"Wert umgewandelt."
110   ZAHL=VAL(Z$)
120   PRINT"Die Zahl lautet nun: ";ZAHL
130   PRINT"Addiert ergibt dies: ";ZAHL+ZAHL
```

Das Beispiel dokumentiert sich selbst. Besonders hinweisen möchte ich noch auf Zeile 110, in der die Umwandlung stattfindet.

Als nächstes soll ein etwas umfangreicheres Beispiel behandelt werden. Es geht dabei im Prinzip um die Prüfung der Eingabe und den sich daraus ergebenden Sprung zu einem bestimmten Programmteil.

Aufgabe:

In einem Menü soll die Eingabe mit dem Befehl VAL geprüft werden und die entsprechenden Reaktionen daraufhin eingeleitet werden.

Eingabe:

Gewünschtes Kennzeichen

Verarbeitung:

Kennzeichen in numerischen Wert umwandeln und prüfen. Wenn richtig, dann Unterprogramm aufrufen, sonst Fehlermeldung ausgeben.

Ausgabe:

Welcher Programmteil wurde aufgerufen

Hinweis:

Zur Erinnerung noch kurz die Arbeitsweise des Menüs.

```
                 H A U P T M E N U E
                 ======================

       1 = Flächenberechnung

       2 = Volumenberechnung

       3 = Gewichtsberechnung

       -----------------------------------------
       Bitte waehlen Sie eine Funktion (1-3)  :
```

Sie brauchen also nur noch die Kennziffer einzugeben und das Programm weiß dann, was Sie möchten.

Das Programm:

```
10    MODE 1
20    INK 0,0
30    BORDER 0
40    CLS
50    PRINT"                 H A U P T M E N U E "
60    PRINT"               ========================"
70    PRINT
80    PRINT"            1 = Flächenberechnung"
90    PRINT
100   PRINT"            2 = Volumenberechnung
110   PRINT
120   PRINT"            3 = Gewichtsberechnung
130   PRINT
140   PRINT"----------------------------------------"
150   PRINT"Bitte waehlen Sie eine Funktion (1-3)  :  "
160   EINGABE$=INKEY$: IF EINGABE$="" THEN 160
170   IF VAL(EINGABE$)=1 THEN GOTO 1000
180   IF VAL(EINGABE$)=2 THEN GOTO 2000
190   IF VAL(EINGABE$)=3 THEN GOTO 3000
200   PRINT"Falsche Eingabe!  Nochmal."
210   FOR PAUSE=1 TO 2000:NEXT PAUSE
220   GOTO 10
1000  PRINT"Hier beginnt der Programmteil Flächenberechnung"
1900  FOR PAUSE=1 TO 1000:NEXT PAUSE:GOTO 10
2000  PRINT"Hier beginnt der Programmteil Volumenberechnung"
2900  FOR PAUSE=1 TO 1000:NEXT:GOTO 10
3000  PRINT"Hier beginnt der Programmteil Gewichtsberechnung"
3900  FOR PAUSE=1 TO 1000:NEXT:GOTO 10
```

Programmbeschreibung:

Bis zur Zeile 150 ist im Prinzip schon alles bekannt; hier wird nur der Bildschirm entsprechend eingestellt und das Menü mit `PRINT` ausgegeben. Zeile 160 fragt die Tastatur ab. Wenn nichts eingegeben wird, wartet das Programm auf einen Tastendruck des Anwenders. In den folgenden Zeilen wird die getätigte Eingabe in einen numerischen Wert umgewandelt und geprüft, ob es sich bei den numerischen Daten um eine 1, 2 oder 3 handelt. Wenn ja, verzweigt das Programm zu dem entsprechenden Programmteil und druckt eine Meldung aus. Bei allen anderen Eingaben erzeugt es eine Fehlermeldung und geht dann wieder zum Anfang zurück.

Verbesserungsmöglichkeiten:

Stellen Sie sich vor, Sie hätten 9 Punkte im Menü und müßten jeden einzelnen Punkt abfragen mit `IF VAL(EINGABE$) = ..... THEN`

`GOTO Zeilennummer`. Nach unserem bisherigen Wissen würde dies 9 Abfragen ergeben. Eine wesentlich einfachere Möglichkeit der Abfrage ist folgende:

```
170 ON VAL(EINGABE$)GOTO 1000,2000,3000
```

Setzen Sie diese Zeile einmal ein, und löschen Sie stattdessen die Zeilen 180 und 190, die Sie dann nicht mehr benötigen. Mit diesem Befehl können Sie platzsparender arbeiten, weshalb Sie diese Art der Abfrage auch beibehalten sollten.

5.24 Der Befehl LEN

Dieser Befehl berechnet die Länge einer Zeichenkette. Als Beispiel dient das folgende Programm.

```
10    CLS:INPUT"Bitte geben Sie einen beliebigen Begriffein:    ",B$
20    LAENGE=LEN(B$)
30    PRINT"Der Begriff: ";B$;" hat eine Länge von";LAENGE;" Zeichen."
```

5.25 Der Befehl SPACE$

Dieser Befehl ist recht nützlich, wenn man beispielsweise Überschriften erstellen möchte und darin größere Leerräume benötigt.

Beispiel:

```
10    MODE 1
20    CLS
30    A$="Laenge" + SPACE$(9) + "Breite" + SPACE$(9) + "Flaeche"
40    PRINT A$
50    PRINT STRING$(40,154)
60    FOR I=1 TO 9
70    PRINT I;SPACE$(13);I;SPACE$(13);I*I
80    NEXT I
90    PRINT STRING$(40,154)
```

Da Sie einiges an Speicherplatz sparen, ist die Verwendung des `SPACE$`-Befehls für größere Leerräume sehr empfehlenswert.

5.26 Die Befehle UPPER$ und LOWER$

Oft ist es nötig, Eingaben in Großbuchstaben oder in Kleinbuchstaben umzuwandeln. So ist es bei Prüfungen zum Beispiel recht nützlich, wenn die Eingabe komplett umgewandelt wird. Ein Beispiel dazu:

```
10    MODE 1
20    CLS
30    INPUT"Bitte Text eingeben:  ",eingabe$
40    PRINT"Sie haben den Text ";CHR$(24);EINGABE$;CHR$(24);" eingegeben."
50    EINGABE$=UPPER$(EINGABE$)
60    PRINT"Umgewandelt sieht Ihr Text so aus: ";CHR$(24); EINGABE$;CHR$(24)
```

Dieses Programm macht aus allen eingegebenen Buchstaben große Buchstaben. Die Umkehrfunktion lautet:

```
50 EINGABE$=LOWER$(EINGABE$)
```

5.27 Die Befehle GOSUB und RETURN

Der BASIC-Befehl **GOSUB** ruft ein sogenanntes Unterprogramm auf. Sicherlich werden Sie sich jetzt fragen, was ein Unterprogramm überhaupt ist?

Stellen Sie sich vor, in einem Programm tauchen an verschiedenen Stellen immer wieder die gleichen Programmzeilen auf, was beispielsweise bei einer Warteschleife der Fall sein könnte. Sehen Sie sich dazu das folgende Beispiel an:

```
10    PRINT"eins"
20    FOR PAUSE=1 TO 1000
30    NEXT PAUSE
40    PRINT"zwei"
50    FOR PAUSE=1 TO 1000
60    NEXT PAUSE
70    PRINT"drei"
80    FOR PAUSE=1 TO 1000
90    NEXT PAUSE
100   PRINT"Ende"
```

Wie Sie sehen, folgt hier nach jedem Ausdruck eine Warteschleife im Programm-Listing. Diese Warteschleife soll jetzt ans Programm-Ende gesetzt und später jedesmal mit dem Befehl **GOSUB** aufgerufen werden.

Das zweite Programm:

```
10    PRINT"eins"
20    GOSUB 1000
30    PRINT"zwei"
```

```
40    GOSUB 1000
50    PRINT"drei"
60    GOSUB 1000
70    PRINT"Ende"
80    END
100   FOR PAUSE=1 TO 1000:NEXT PAUSE:RETURN
130   PRINT"Ende des Programms"
```

Die Ausgaben mit PRINT bleiben, wie bereits im ersten Programm, gleich. Statt der Schleife hinter den PRINT-Befehlen steht jetzt jedoch immer der Aufruf eines Unterprogramms. Mit dem Befehl GOSUB 1000 wird dieses Unterprogramm mehrfach aus dem Hauptprogramm aufgerufen.

Das Programm verzweigt also nach 1000 und arbeitet dort die Warteschleife ab. Anschließend stößt es auf den Befehl RETURN, der es wieder zurückschickt. Eine Besonderheit bei dem Befehl RETURN ist, daß keine direkte Zeilennummer für den Rücksprung angegeben wurde. Vielmehr merkt sich das Programm die Zeilennummer, von der es in das Unterprogramm gesprungen ist. Mit dem Befehl RETURN springt das Programm wieder zurück zu demjenigen Befehl, der gleich nach dem Aufruf durch GOSUB 1000 im Programm als nächstes erscheint. Deshalb ist auch keine Angabe nötig, wohin es gehen soll.

Bei der eben beschriebenen Warteschleife handelt es sich also um ein Unterprogramm. Ein Unterprogramm ist recht übersichtlich aufgebaut und kann von mehreren Stellen im Programm aufgerufen werden. Der besseren Übersicht halber, setzt man solche Unterprogramme ans Ende eines Programms.

Noch ein Tip: Wählen Sie für Ihre Unterprogramme möglichst gerade Zeilennummern, weil Sie sich diese beim Programmieren besser merken können. Ich persönlich würde beispielsweise niemals folgendes schreiben:

```
11287 FOR P=1 TO 1000: NEXT: RETURN
```

Eine solche Zeile sähe bei mir so aus:

```
11300 FOR PAUSE=1 TO 1000: NEXT PAUSE: RETURN
```

Innerhalb eines Unterprogramms können weitere Unterprogramme aufgerufen werden. Je mehr Sie davon verwenden, desto übersichtlicher sind Ihre Programme. Ich erstelle Unterprogramme auf folgende Weise:

Zuerst zerlege ich ein geplantes Programm in einzelne Bausteine. Anschließend wird Baustein für Baustein als Unterprogramm erstellt und separat ausgetestet. Funktioniert ein Baustein einwandfrei, wird er sofort

abgespeichert und das Programm-Listing kommt in die Baustein-Bibliothek.

Benötigt man dann später einmal einen bestimmten Baustein, kann man zuerst in der Programmbibliothek nachsehen, ob er dort bereits vorhanden ist. Falls ja, kann man ihn in das neue Programme integrieren, wobei man allerdings auf die richtige Numerierung der Zeilen achten muß. Da man viele kleine Routinen und andere Bausteine Prinzip immer wieder benötigt, ist es recht hilfreich, wenn man auf vorhandene zurückgreifen kann.

In dem Kapitel "Fertig programmierte Programmbausteine" finden Sie einen Auszug aus meiner persönlichen Bibliothek für den CPC 464.

Nutzen Sie also diese Bausteine, da Sie damit eine Menge Zeit sparen und sich stattdessen auf das eigentliche Problem konzentrieren können.

Beispiel 1: Rabattberechnung

Aufgabe:

Es soll ein Programm entwickelt werden, das ab einer bestimmten Summe einen vorgegebenen Rabatt vom Gesamtbetrag abzieht.

Die Ausgangswerte:

Unter 900 DM = 2% Rabatt
Über 900 DM = 5% Rabatt

Eingabe:

Gesamtbetrag in DM

Verarbeitung:

Prozentsatz für den Rabatt festlegen
Rabatt in DM errechnen
Endbetrag errechnen

Ausgabe:

Gesamtbetrag ohne Rabatt
Rabatt in %
Rabatt in DM
Betrag mit Rabatt in DM

Das Programm:

```
10    MODE 1:CLS:INK 0,0:BORDER 0
20    PRINT"R A B A T T - B E R E C H N U N G"
```

```
30    PRINT"================================="
40    PRINT
50    PRINT"Bitte geben Sie den Gesamtbetrag in DM ein "
60    INPUT GESAMT
70    REM **************Rabattberechnung einleiten *******
80    IF GESAMT < = 900 THEN PROZENT=2:GOSUB 1000
90    IF GESAMT > 900 THEN PROZENT=5:GOSUB 2000
95    ENDBETRAG = GESAMT - DM
100   REM ************ Datenausgabe auf Bildschirm ********
110   CLS
120   PRINT"Gesamtbetrag:        ";TAB(25);GESAMT
130   PRINT"Rabatt in Prozent:   ";TAB(25);PROZENT
140   PRINT"Rabatt in DM:        ";TAB(25);DM
150   PRINT"Endbetrag in DM:     ";TAB(25);ENDBETRAG
160   END
1000  REM ****** Unterprogramme für 2% Rabatt ***********
1010  DM=GESAMT/100*2
1020  RETURN
2000  REM ****** Unterprogramm für 5 % Rabatt ***********
2010  DM=GESAMT/100*5
2020  RETURN
```

5.28 Die Befehle OPENOUT und CLOSEOUT

Mit den beiden Befehlen OPENOUT und CLOSEOUT können Sie Daten auf einem externen Speicher abspeichern. Daten sind z.B. Eingaben oder Berechnungen, die innerhalb eines Programmes gemacht wurden. Da beim Ausschalten des Computers alles verloren geht, kann man die Daten auf Kassette oder Diskette abeichern und sie somit jederzeit wieder in den Speicher des Computers holen.

Man spricht bei einer solchen Speicherung von einer Datei, in die die einzelnen Daten geschrieben werden. Damit der Computer die einzelnen Dateien auseinander halten kann, benötigt jede Datei einen Namen. Dieser Name darf bei der Verwendung von Kassetten 16 Zeichen und bei der Verwendung von Disketten nur 8 Zeichen lang sein. Als Dateinamen sollte man möglichst aussagekräftige Namen wählen, damit man später auch Rückschlüsse auf den Inhalt der jeweiligen Datei ziehen kann.

Beispiel: Sie möchten eine Datei für Adressen anlegen. Ein angemessener Dateiname wäre dann: ADRESSEN

Sie werden sich jetzt sicherlich fragen, wie denn diese Daten überhaupt gespeichert werden. Das Prinzip, das ich Ihnen nachfolgend am Beispiel darlegen möchte, ist bei jeder Dateiverwaltung gleich.

1. Schritt: Datei mit dem Namen .... öffnen zum Schreiben

2. Schritt: Gewünschte Daten in die Datei schreiben
3. Schritt: Datei wieder schließen

Die einzelnen Schritte lauten in BASIC konkret:

```
10    OPENOUT"ADRESSEN"
20    PRINT#9,.......
30    CLOSEOUT
```

Eine Datei zum Speichern von Daten muß also zuerst mit dem jeweiligen Dateinamen eröffnet werden, bevor Sie sie anderweitig benutzen können. Der nächste Schritt ist das Speichern der gewünschten Daten. Hier spricht man in der Fachsprache nicht mehr vom Speichern, sondern vom Schreiben der Daten.

Mit dem Befehl **PRINT#9** werden auf einen externen Speicher die gewünschten Daten geschrieben. Die 9 ist dabei die Gerätenummer für den Kassettenrekorder oder, falls angeschlossen, für das Diskettenlaufwerk.

Trifft das Programm irgendwann auf den Befehl **OPENOUT**, erscheint eine Meldung auf dem Bildschirm, in der Sie aufgefordert werden, die Tasten <PLAY> und <RECORD> am Rekorder zu drücken. Danach muß als Bestätigung noch eine Taste auf der Tastatur gedrückt werden, und schon beginnt der Schreibvorgang. Bei Verwendung des Diskettenlaufwerks entfällt diese Meldung und es wird sofort auf die Diskette geschrieben.

Unterdrücken kann man diese Meldung beim Kassettenrekorder, indem man ein Ausrufungszeichen vor den Dateinamen setzt. Beispiel: Sie wollen die Systemmeldung unterdrücken und schreiben dazu:

```
OPENOUT "!ADRESSEN"
```

Der Nachteil dieses Vorgehens ist, daß Sie nichts mehr auf dem Bildschirm sehen. Es gibt jedoch auch hier eine recht elegante Möglichkeit, den Text dennoch sichtbar zu machen: Schreiben Sie vor dem **OPENOUT** einfach mit einer **PRINT**-Zeile eine Nachricht auf den Bildschirm.

Es folgt ein Beispiel, wie man Daten, in diesem Fall sind es Zufallszahlen, auf einen Externspeicher schreiben kann.

```
10    REM ***************************************************
20    REM **                                               **
30    REM **   Demonstrationsprogramm zur Speicherung     ***
40    REM **                                              ***
50    REM ** von numerischen Daten beim Schneider CPC 464 **
60    REM **                                               **
70    REM **                                               **
```

```
80    REM ****************************************************
90    MODE 2:CLS:INK 0,0: BORDER 0
100   REM
110   REM ********* 9 zufällige Zahlen erzeugen ************
120   FOR I=1 TO 9
130   Z(I)=INT(RND(1)*1000)
140   NEXT I
150   REM ********** Ausgabe auf Bildschirm ****************
160   FOR I=1 TO 9
170   PRINT I;".te Zahl lautet  ";Z(I)
180   NEXT I
190   REM ****************** Datenspeicherung beginnt ******
200   PRINT"Bitte drücken Sie die Tasten REC und PLAY an Ihrem"
210   PRINT"Rekorder, wenn Sie kein Diskettenlaufwerk haben"
220   OPENOUT"!testdaten"
230   FOR I=1 TO 9
240   PRINT#9,Z(I)
250   NEXT I
260   CLOSEOUT
270   PRINT"Daten alle in die Datei geschrieben...."
280   END
```

Dieses Beispiel erzeugt Ihnen neun Zufallszahlen und gibt sie auf dem Bildschirm aus. Anschließend werden Sie aufgefordert, die beiden Tasten zur Aufnahme am Kassettenrekorder zu drücken. Sollten Sie über ein Diskettenlaufwerk verfügen, erübrigt sich dies natürlich.

Mit dem Befehl PRINT#9 werden nun alle Zufallszahlen in die Datei mit dem Namen TESTDATEN geschrieben.

```
10    REM *****************************************************
20    REM **                                                 **
30    REM **  Demonstrationsprogramm zum Speichern           **
40    REM **                                                 **
50    REM **  von alphanum. Daten beim Schneider CPC 464     **
60    REM **                                                 **
70    REM **                                                 **
80    REM *****************************************************
100   MODE 2:CLS:INK 0,0:BORDER 0
110   REM
120   FOR I=1 TO 3
130   CLS:PRINT"Bitte eine komplette Anschrift hier erfassen.."
140   INPUT"Vorname";VORNAME$(I)
150   INPUT"Nachname";NACHNAME$(I)
160   INPUT"Strasse und Nr.: ";STRASSE$(I)
165   INPUT"PLZ und Ort      ";ORT$(I)
170   INPUT"Telefonnummer    ";TELEFON$(I)
180   NEXT I
200   REM *************** Datenspeichern beginnt **************
210   OPENOUT "testadressen"
220   FOR I=1 TO 3
230   PRINT#9,VORNAME$(I)
231   PRINT#9,NACHNAME$(I)
232   PRINT#9,STRASSE$(I)
233   PRINT#9,ORT$(I)
```

```
234    PRINT#9,TELEFON$(I)
240    NEXT I
250    CLOSEOUT
260    PRINT"Daten sind jetzt gespeichert...."
```

In diesem Beispiel werden keine Zahlen, sondern Anschriften auf einen Externspeicher geschrieben. Dabei werden drei komplette Anschriften eingegeben, die mit PRINT#9 in eine Datei mit dem Namen TESTADRESSEN geschrieben werden.

Falls Sie ein Diskettenlaufwerk besitzen, sollten Sie sich einen Namen auswählen, der nur acht Buchstaben lang ist.

Abschließend zur Wiederholung nochmals die einzelnen Schritte bei der Dateneingabe:

1. Schritt: Öffnen der Datei
2. Schritt: Schreiben in die Datei
3. Schritt: Schließen der Datei

5.29 Die Befehle OPENIN und CLOSEIN

Hier trifft im Prinzip all das zu, was bereits im letzten Kapitel gesagt wurde. Mit diesen beiden Befehlen werden die Daten wieder vom Externspeicher zurück in den Arbeitsspeicher des Computers gelesen, damit sie zur weiteren Verarbeitung zur Verfügung stehen.

Im letzten Kapitel wurden zwei Programme besprochen, die Daten auf eine Kassette oder Diskette gespeichert haben. Hier nun die beiden Programme, die diese Daten wieder lesen können.

Achten Sie bei Ihren eigenen Programmen immer darauf, daß dieselbe Reihenfolge beim Lesen eingehalten wird, wie beim Schreiben! Ansonsten kann es zu Problemen beim Lesen kommen, mit denen man vorher nicht gerechnet hat.

```
10     REM ******************************************************
20     REM **                                                  **
30     REM **  Demonstrationsprogramm zum Laden                **
40     REM **                                                  **
50     REM **  von numerischen Daten beim Schneider CPC 464    **
60     REM **                                                  **
70     REM **                                                  **
90     REM ******************************************************
100    MODE 2:CLS:INK 0,0:BORDER 0
110    REM
200    REM ********** Datenladen beginnt ***********************
```

```
210    OPENIN"testdaten"
220    FOR I=1 TO 9
230    INPUT#9,Z(I)
240    NEXT I
250    CLOSEIN
260    PRINT"Daten alle aus der Datei gelesen....."
360    REM ********** Ausgabe auf Bildschirm *****************
370    FOR I=1 TO 9
380    PRINT I;".te Zahl lautet:  ";Z(I)
390    NEXT I
400    END

10     REM *****************************************************
20     REM **                                                 **
30     REM **  Demonstrationsprogramm zum Laden               **
40     REM **                                                 **
50     REM **  von alphanum. Daten beim Schneider CPC 464     **
60     REM **                                                 **
70     REM **                                                 **
80     REM *****************************************************
90     MODE 2:CLS:INK 0,0:BORDER 0
100    REM
110    REM ******* Datenladen beginnt .........   **************
120    OPENIN"testadressen"
130    FOR I=1 TO 3
140    INPUT#9,VORNAME$(I)
150    INPUT#9,NACHNAME$(I)
160    INPUT#9,STRASSE$(I)
170    INPUT#9,ORT$(I)
180    INPUT#9,TELEFON$(I)
190    NEXT I
200    CLOSEIN
210    PRINT"Daten alle aus der Datei gelesen..........."
220    REM *************** Ausgabe auf Bildschirm ************
230    FOR I=1 TO 3
240    PRINT"Vorname ";VORNAME$(I)
250    PRINT"Nachname";NACHNAME$(I)
260    PRINT"Strasse ";STRASSE$(I)
270    PRINT"Ort     ";ORT$(I)
280    PRINT"Telefon ";TELEFON$(I)
290    PRINT"=============================="
300    PRINT
310    NEXT I
```

5.30 Der Befehl EOF

Mit diesem Befehl wird das Ende einer Datei abgefragt. Dabei bedeutet EOF: END OF FILE.

Sinnvoll ist dieser Befehl bei Dateien, deren Größe ständig schwankt und daher kein feste Anzahl von einzulesenden Datensätzen hat. Eine Datei mit schwankender Anzahl Datensätze könnte z.B. so eingelesen werden:

```
10    OPENIN"Test"
20    WHILE NOT EOF
30    INPUT#9, T
40    WEND
50    CLOSEIN
```

Es wird solange die Variable T mit Daten gefüllt, bis keine mehr da sind. Dann verläßt das Programm automatisch die Schleife und schließt die Datei wieder.

5.31 Der Befehl CHAIN

Beim Programmieren in BASIC stehen Ihnen im Normalfall 42 KByte an Arbeitsspeicher zur Verfügung. Theoretisch dürfen Ihre Programmentwicklungen auch nicht mehr als diesen Speicherplatz verbrauchen.

Mit dem CHAIN-Befehl ist es möglich, Teile eines sehr großen Programmes zu laden und abzuarbeiten. Angenommen, Ihr Programm verbraucht unter normalen Umständen rund 80 KByte an Speicherplatz. Teilen Sie es in kleinere Einheiten auf und laden Sie dann Teil für Teil in den Arbeitsspeicher, wo es abgearbeitet wird. Am Schluß eines Teilprogramms muß dann wieder der Befehl CHAIN stehen.

Beispiel:

```
10    REM ************************************
20    REM ***** das ist der erste Teil ******
30    .....
40    .....
5000  REM ** hier hört der erste Teil auf ***
5010  CHAIN"Test",5020
```

Sie starten das Programm mit RUN. Daraufhin werden die Zeilen bis 5000 abgearbeitet. Dann wird das Programm mit dem Namen "Test" von der Kassette oder Diskette geladen, wobei das geladene Programm von der neuen Programmnummer (5020) an weiterarbeitet.

Der Vorteil von CHAIN ist, daß alle Variablen mit in das nächste Programm übernommen werden!

Der zweite Teil unseres Beispielprogramms könnte dann so aussehen:

```
5020  REM ********* Zweiter Teil des Programms ******
5030  ....
5040  ....
9000  REM *** hier hört der zweite Teil auf *********
9010  CHAIN"Ausgabe",10
```

Hier lädt das zweite Programm nach seiner Beendigung das Programm "Ausgabe" und beginnt dort mit Zeile 10.

5.32 Der Befehl MERGE

Sehr häufig kommt es vor, daß man Programme von Kassette oder Diskette nachladen möchte, ohne dabei das alte Programm im Speicher zu löschen.

Mit dem Befehl MERGE ist das Mixen vom Programmen kein Problem mehr. Angenommen, Sie haben zwei Programme, die Sie miteinander mischen möchten. Das erste Programm befindet sich bereits im Arbeitsspeicher des Computers und sieht folgendermaßen aus:

```
10    REM  ---------- Demo -----------
20    REM
30    REM
40    REM  ---------- Ende -----------
```

Den zweiten Teil des Programms wollen Sie von einem Externspeicher (Kassette oder Diskette) zusätzlich in den Speicher bringen.

Dazu geben Sie den Befehl MERGE und anschließend den neuen Programmnamen ein:

```
MERGE"Ausgabe"
```

Das geladene Programm besitzt die Zeilen:

```
100   PRINT"Druckerausgabe"
110   PRINT"=============="
120   PRINT
130   PRINT"Ende dieses Programms"
```

Jetzt werden beide Programme im Arbeitsspeicher des Computers gemischt und Sie erhalten das folgende Listing:

```
10    REM  ---------- Demo ------
20    REM
30    REM
40    REM...---------- Ende ------
100   PRINT"Druckerausgabe"
```

```
110   PRINT"=============="
120   PRINT
130   PRINT"Ende dieses Programms"
```

6 Speichern und Laden von Programmen

Schon beim Kauf haben Sie bemerkt, daß Ihr Schneider einen eingebauten Kassettenrekorder besitzt. Dieser dient nicht etwa dazu, neben der Arbeit noch Musik abzuspielen, sondern er wird als Datenspeicher gebraucht. Vielleicht werden Sie sich fragen, wie es möglich ist, daß Computerprogramme auf eine ganz normale Kassette gelangen. In der nun folgenden Beschreibung erhalten Sie dafür die Erklärung.

In jedem Kassettenrekorder sind sogenannte Schreib- und Leseköpfe eingebaut, die sich meist in einem einzigen Gehäuse befinden. Nehmen wir einmal an, ein Computerprogramm soll auf Band gespeichert werden. Wird für die Aufnahme die Tastenkombination <RECORD> und <PLAY> am Kassettenrekorder gedrückt, fährt der Schreib- und Lesekopf ganz nah an das Band der eingelegten Kassette heran. Nun beginnt die Datenübertragung auf Band. Das Programm im Speicher des Computers wird dazu in zwei verschiedene Tonsignale umgewandelt.

Damit diese beiden Signale auch von einem Magnetaufzeichnungsgerät verarbeitet werden können, hat man sich darauf geeinigt, daß sie aus zwei unterschiedlichen Tonhöhen bestehen. Diese Töne können wie eine Musikaufnahme auf Kassette gespeichert werden. Mit dem Schreibkopf des Rekorders werden sie in unterschiedliche Magnetimpulse umgewandelt und auf das vorbeilaufende Band der Kassette geschrieben. Am Anfang sendet der Computer immer ein Signal, das als Vorspann gebraucht wird. Es dient als Kennzeichnung für den Anfang und das Ende eines Programms.

Nach dem normalen Vorspann folgt das eigentliche Programm. Wenn Sie beim Speichern die Lautstärke am Schneider aufdrehen, können Sie die einzelnen Phasen der Übertragung gut verfolgen. Wenn das Speichern des Programms beendet ist, meldet sich der Computer automatisch mit READY und schaltet den Motor des Rekorders ab.

Sie haben jetzt ein Computerprogramm auf einer Kassette gespeichert und können es jederzeit wieder in den Computer laden. Dies ist wichtig, weil Sie ja irgendwann den Rechner ausschalten wollen und dabei unweigerlich

alle Eingaben verloren gehen. Deshalb sichert man sich Programme auf Band.

Beim Laden eines Programms wiederholt sich die ganze Prozedur in umgekehrter Reihenfolge. Jetzt werden die Magnetsignale wieder gelesen und in ein Computerprogramm zurückverwandelt. Somit haben Sie die Möglichkeit, Programme auf handelsüblichen Leerkassetten zu sichern und sie beliebig oft zu laden.

6.1 Der Befehl SAVE

Mit dem Befehl **SAVE** können Sie alle Ihre Programme auf einem externen Speichermedium speichern (sichern). Die gängigsten Speichermedien sind zur Zeit die Kassette und die Diskette. Der Befehl **SAVE** wird folgendermaßen angewendet:

```
SAVE"Name des Programms"
```

Für jedes Ihrer Programme brauchen Sie einen Namen, um darauf beim Laden gezielt zurückgreifen zu können. Bei der Vergabe von Namen sollten Sie sich möglichst Begriffe wählen, die einen Bezug zum Programm haben. Zum Beispiel könnte man ein Programm, das Flächen berechnet, unter dem Name "FLÄCHE" abspeichern. Wenn Sie ein Programm ohne Namen abspeichern, besteht später nicht mehr die Möglichkeit, es auf Anhieb auf der Kassette zu finden.

Mit dem Befehl **SAVE** können Sie also Ihre geschriebenen Programme auf einer Kassette sichern. Dabei gehen Sie zweckmäßigerweise so vor:

1. Suchen Sie sich einen Programm-Namen aus, der aber 16 Zeichen nicht überschreiten darf. Bei mehr als 16 Zeichen werden nur die ersten 16 für den Programm-Namen verwendet.

2. Machen Sie Ihre Kassette im Rekorder startklar.

3. Geben Sie nun den Befehl `SAVE"PROGRAMM-NAME"` ein und drücken Sie zur Bestätigung die Taste <ENTER>.

4. Sie erhalten nun die Meldung auf dem Bildschirm:

```
Press REC and PLAY then any key
```

Diese Nachricht bedeutet für Sie, daß nun gleichzeitig die beiden Tasten <REC> und <PLAY> am Rekorder gedrückt werden müssen. Anschließend betätigen Sie dann eine beliebige andere Taste.

5. Wenn Sie alles richtig gemacht haben, läuft nun der Motor des Rekorders an und das entsprechende Programm wird gespeichert.

6. Ist der Speichervorgang beendet, bleibt auch der Motor des Rekorders stehen. Sie können jetzt durch Drücken der Taste <STOP/EJECT> die beiden eben gedrückten Tasten (<REC> und <PLAY>) wieder in die Ausgangsposition zurücksetzen.

7. Ihr Programm ist nun auf Kassette gespeichert. Merken Sie sich am besten noch den Stand des Zählwerks, und notieren Sie ihn auf der Kassette.

Ich kann Ihnen aus eigener Erfahrung berichten, welches Chaos entsteht, wenn Programmkassetten nicht ausreichend beschriftet werden. Deshalb sollten Sie am besten gleich nach dem Speichern die Kassetten beschriften oder einen kleinen Zettel mit Angaben zum Programm in die Kassettenbox legen.

Programme können mit einem bestimmten Format abgespeichert werden, wenn Sie dem Rechner dies am Ende des SAVE-Befehls mitteilen. Wie so etwas funktioniert und welche Formate der Schneider zur Verfügung stellt, werden wir im folgenden Abschnitt behandeln.

Mögliche Datenformate beim Speichern

Wie bereits im letzten Abschnitt erwähnt, gibt es verschiedene Möglichkeiten, Programme mit SAVE abzuspeichern. Dabei unterscheidet man 4 Möglichkeiten:

1. Abspeichern mit dem Befehl SAVE ohne einen Zusatz.

   Beispiel: `SAVE"RECHNUNG`

2. Abspeichern im sogenannten ASCII-Code

   Beispiel: `SAVE"RECHNUNG",A`

3. Abspeichern mit LIST-Schutz

   Beispiel: `SAVE"RECHNUNG",P`

4. Abspeichern im binären Datenformat

Beispiel: `SAVE "RECHNUNG",B,STARTADRESSE,LÄNGE`

Die erste Möglichkeit haben wir schon kennengelernt und brauchen darauf nicht mehr einzugehen.

Die zweite Möglichkeit speichert Programme im sogenannten ASCII-Code ab. Der ASCII-Code wird bei Verwendung des normalen SAVE-Befehls in einen für den Computer lesbaren Maschinencode umgewandelt. Oft ist es aber wünschenswert, Programme, oder Teile von Programmen, zu bearbeiten, die sich schon auf Kassette befinden. Denkbar wäre z.B. eine Übernahme von Programmen in eine Textverarbeitung. Da der Schneider aber beim Abspeichern diese Programme in einen anderen Code umwandelt, kann man diese Programme so nicht gebrauchen.

Wird jedoch ein Programm mit dem Zusatz ",A" gespeichert, dann sichert der Schneider Ihnen Ihr Programm genauso, wie Sie es auf dem Bildschirm als Listing gesehen haben. Eine Umwandlung findet dabei nicht statt.

Programme, die als ASCII-Programme abgespeichert werden, verbrauchen mehr Platz auf der Kassette als normal abgespeicherte Programme. Dafür können Sie sie aber als Listings in Textverarbeitungen übernehmen.

Die dritte Möglichkeit, Programme auf Kassette zu speichern, besteht in dem Zusatz ",P". Dabei steht der Buchstabe P für den Begriff *Protect.* Alle Programme, die mit diesem Zusatz abgespeichert werden, lassen sich später nicht mehr auflisten. ",P" verwendet man als Programmschutz, damit fremde Personen das BASIC-Programm nicht verändern können.

Vorsicht ist aber auch hier geboten. Programme, die Sie mit diesem Zusatz abspeichern wollen, sollten Sie zweckmäßigerweise vorher auf einer Kassette normal abspeichern, denn auch Sie selbst können diese Programme später nicht mehr verändern, d.h. Sie können sie weder listen noch eine Zeilennummer hinzufügen. Bitte bedenken Sie das, bevor Sie diese Art der Speicherung wählen!

An dieser Stelle möchte ich ein paar Worte über "Raubkopien" sagen. Raubkopien sind Programme, die jemand unrechtmäßig erworben hat. Vielen ist vielleicht gar nicht bewußt, welcher Schaden damit angerichtet werden kann. Programme lassen sich nicht von heute auf morgen entwickeln. Für manche Projekte braucht man oft Monate, bis sie endlich fertig sind und richtig laufen. Bis dahin hat der Programmierer viel Zeit und Geld investiert, das er natürlich auch wieder verdienen möchte. Durch die unerlaubte Vervielfältigung und Veräußerung von fremden Programmen schaden sich die jeweiligen Anwender zum guten Schluß

selbst, denn keiner wird mehr qualitativ hochwertige Software erstellen, wenn er sie ein halbes Jahr später in fast jeder Tauschliste von Computerfreunden wiederfindet. Außerdem sind gute Programme immer mit einer ausführlichen Bedienungsanleitung versehen, die bei einer Raubkopie meist fehlt.

Wird jemand als Raubkopierer entdeckt, reagieren deutsche Gerichte mit Beschlagnahmungen, Gerichtsverhandlungen und empfindlichen Geldbußen. Überlegen Sie sich also, ob sich eine solche Handlungsweise letztendlich wirklich lohnt?!

Mit der vierten Möglichkeit können Sie Programme so abspeichern, wie sie im Computer stehen. In einem solchen Fall spricht man von binärer Speicherung. Mit dem Zusatz ",B,Anfangsadresse, Länge," können beispielsweise vom Bildschirm Abzüge gemacht und diese auf Kassette gespeichert werden. Mit dem folgenden Programm können Sie den gesamten Inhalt des Bildschirms auf Kassette überspielen.

```
10    MODE 1
20    CLS
30    FOR I=1 TO 200:PRINT"#";:NEXT
40    SAVE"TESTBILD",B, &C000, &4000
```

Mit einem LOAD-Befehl werden solche Bilder später in andere Programme wieder eingebunden. Das eben vorgestellte Beispiel zeichnet Ihnen 200 Zeichen auf den Bildschirm und speichert anschließend den Bildschirminhalt ab.

6.2 Speichern mit doppelter Geschwindigkeit

Wenn Sie möchten, können Sie beim Speichern auch die doppelte Geschwindigkeit zum Aufzeichnen verwenden. Dies geschieht mit dem Befehl SPEED WRITE.

Im normalen Zustand zeichnet der Schneider mit einer Übertragungsrate von 1000 Baud auf. 1000 Baud bedeuten eine Übertragungsgeschwindigkeit von 1000 Zeichen pro Sekunde, mit der die Daten auf die Kassette gespeichert werden. Mit dem BASIC-Befehl SPEED WRITE 1 schaltet der Rechner das Aufzeichnungsformat von 1000 auf 2000 Baud um. Damit erhöht sich die Speicher- und Ladegeschwindigkeit um 100 %. Besonders bei längeren Programmen ist diese Einrichtung recht angenehm, denn wer will schon ewig warten, bis seine Programme endlich im Speicher sind.

Wenn Programme mit doppelter Geschwindigkeit gespeichert worden sind, erkennt der Computer beim Laden eines Programms automatisch die jeweilige Aufzeichnungsgeschwindigkeit und schaltet dementsprechend auf das aktuelle Format um.

Einen Nachteil hat die doppelte Aufzeichungsgeschwindigkeit jedoch: Da die Informationen jetzt noch gebündelter auf dem Magnetband liegen, kommt es häufiger zu Lesefehlern, als bei der Übertragungsgeschwindigkeit von 1000 Baud. Kommt es wegen der höheren Informationsdichte zum sogenannten "*Load error*", haben Sie hoffentlich noch eine Kopie Ihres Programms irgendwo im Schrank liegen, sonst ist Ihr Programm u.U. für Sie verloren. Besonders häufig treten solche Fehler bei der Benutzung minderwertiger Kassetten auf. Wenn dann noch mit `SPEED WRITE 1` gespeichert wird, sind derartige Fehler schon fast vorprogrammiert.

Deshalb mein Tip: Speichern Sie ruhig alle Programme mit `SPEED WRITE 1` ab, legen Sie sich aber eine Kopie mit einem Aufzeichnungsformat von 1000 Baud an, die Sie an einem sicheren Ort deponieren. Noch besser ist es natürlich, wenn Sie sich von allen wichtigen Programmen zwei Kopien anfertigen. So haben Sie für den Ernstfall auf jeden Fall ein Duplikat des Originals, mit dem Sie dann weiterarbeiten können. Der Ernstfall tritt oft schneller ein, als einem lieb ist. Plötzlich läßt sich eine sonst einwandfreie Kassette nicht mehr lesen, dann sind Sie für jede Kopie dankbar, die Sie sich gemacht haben.

Wer beispielsweise versucht, bei Bandsalat einfach ein kleines Stück Magnetband wegzuschneiden und das Band neu aneinander zu kleben, wird beim Laden seines Programms mit Sicherheit Schiffbruch erleiden. Anders, als bei Musikkassetten, benötigt der Computer jedes Stückchen Band, das mit Programmen belegt ist, sonst kommt es zu Ladefehlern.

Mit dem Befehl `SPEED WRITE 0` schalten Sie wieder auf 1000 Baud zurück.

6.3 Tips für die Anschaffung von Kassetten

Generell eignet sich jede unbespielte Musikkassette zur Speicherung von Daten. Leider gibt es jedoch oft erhebliche Qualitätsunterschiede bei Kassetten. Nicht zu empfehlen sind solche, die für den gehobenen HIFI-Bereich gedacht sind. Diese Kassetten sind meist zu empfindlich für den Einsatz als Speichermedium für Computer. Sie erkennen sie schon an

Ihrem Preis, da es mit Abstand die teuersten Musikkassetten auf dem Markt sind.

Ebenfalls unbrauchbar und wenig empfehlenswert sind Kassetten, die häufig im Zehnerpack angeboten werden und unglaublich preiswert sind. Bei solchen Angeboten handelt es sich fast immer um Material von minderwertiger Qualität, was später zu Störungen führen kann.

Empfehlenswert sind handelsübliche Chromdioxid-Kassetten, deren Länge bei maximal 30 liegen sollte. Achten Sie darauf, daß Sie keine langen Spielzeiten erwerben. Man neigt zwar dazu, auf eine längere Spielzeit auch mehr Programme zu verteilen, in der Praxis führt aber dieser Umstand zu längeren Suchzeiten beim Laden von bestimmten Programmen. Ich verwende fast ausschließlich Kassetten vom Typ C 20, auf denen man zur Not auch noch mehrere Programme hintereinander abspeichern kann. Diese Kassetten sind im Computerversandhandel oder in den Computershops der Kaufhäuser erhältlich. Man findet darauf Programme wesentlich schneller, weil sich von Anfang an nur einige Stellen speichern lassen.

Und nochmals: Wenn Sie Ihr Programm gespeichert haben, vergessen Sie nicht, Ihre Kassetten zu beschriften! Nichts ist schlimmer, als auf einer Unmenge von Kassetten ein bestimmtes Programm suchen zu müssen. Deswegen sollten Sie sofort nach dem Speichern den Programmnamen und den Stand des Zählwerks vermerken.

6.4 Der Befehl LOAD

Mit diesem Befehl können Sie Ihre auf Kassette gespeicherten Programme wieder in den Arbeitsspeicher des Computers laden und damit arbeiten. Es wurde bereits gesagt, daß es keine Rolle spielt, mit welcher Geschwindigkeit (1000 oder 2000 Baud) aufgezeichnet wurde, da sich der Schneider auf das jeweilige Geschwindigkeitsformat automatisch einstellt. Sie brauchen sich also darum beim Laden Ihrer Programme nicht zu kümmern.

Programme können von Kassette geladen werden, wenn man den Befehl LOAD eingibt. Dieser Befehl hat folgendes Format:

```
LOAD"Programm-Name"
```

oder nur einfach

```
LOAD"
```

Mit dem Befehl LOAD" wird das erste Programm geladen, das der Schneider auf der Kassette findet. Ist es geladen, folgt mit dem nächsten LOAD-Befehl das Programm, das als nächstes auf der Kassette steht. Dieses Vorgehen ist immer dann sinnvoll, wenn man vergessen hat, mit welchem Namen ein Programm gespeichert worden ist.

Mit der Anweisung LOAD"NAME DES PROGRAMMS" können Sie sich ein ganz bestimmtes Programm von der Kassette laden. Dabei wird die Kassette u.U. bis zum Ende durchsucht, wenn das gesuchte Programm nicht darauf gespeichert ist. Jetzt verstehen Sie sicher auch, warum ich Kassetten mit kurzen Laufzeiten empfohlen habe.

Etwas sollten Sie beim Laden noch beachten, bevor Sie mit der Arbeit beginnen. Alle Programme, die sich vorher noch im Rechner befanden, werden beim Laden gelöscht. Nach dem Laden haben Sie nur Ihr neues Programm im Arbeitsspeicher und sonst nichts mehr.

Wir wollen uns nun ein Beispiel ansehen, damit Sie sehen, wie dieser Befehl im einzelnen funktioniert. Angenommen, Sie haben auf Kassette ein Programm unter dem Namen "RECHNER" gespeichert und wollen es wieder in den Arbeitsspeicher laden. Geben Sie dazu ein:

```
LOAD"RECHNER"
```

und drücken Sie anschließend die <ENTER>-Taste. Jetzt erscheint auf dem Bildschirm der folgende Hinweis:

```
Press PLAY then any key
```

Sie werden damit aufgefordert, die Taste <PLAY> am Rekorder und danach eine beliebige Taste auf der Tastatur zu drücken. Nun läuft der Motor des Kassettenrekorders an und das Programm "RECHNER" wird gesucht. Wurde es auf der Kassette gefunden, erscheint unter der Meldung "Press PLAY then any key" die Nachricht:

```
Loading RECHNER block 1
```

Die Angabe "block" möchte ich kurz erklären. Ein Programm wird beim Speichern in einzelne Teile, in die sogenannten Blöcke, unterteilt. Jeder Block hat dabei eine Länge von max. 2 Kbyte. Wenn ein Block geladen wird, dann erscheint die jeweilige Nummer des Blocks auf dem Bildschirm. Nach dem letzten Block meldet sich der Schneider wieder mit READY und das gesuchte Programm befindet sich im Speicher des Computers. Dieses Programm können Sie nun beliebig verändern, z.B. listen oder drucken.

Nicht immer lassen sich Programme einwandfrei laden. Sie bemerken meist zuerst an der Nachricht "*Read Error*", daß etwas danebengegangen sein muß. Vor Ladefehlern ist kein Programm sicher; früher oder später tritt dieser Fehler bestimmt einmal auf. Der nächste Abschnitt wird auf dieses Problem näher eingehen.

6.5 Fehler beim Laden von Kassette

Leider ist das Arbeiten mit der Kassette als Speichermedium nicht ganz problemlos. Zwar ist die Kassette im Vergleich zur Diskette wesentlich billiger, dafür aber auch anfälliger für Fehler, die beim Lesen auftreten können.

Lesefehler können verschiedene Ursachen haben. Nach meinen Erfahrungen mit anderen Systemen kann man Lesefehler in zwei Gruppen einteilen, wobei die erste Gruppe weitaus häufiger auftritt als die zweite.

a) Lesefehler, bedingt durch mechanische Einwirkung
b) Lesefehler, bedingt durch starke Magnetfelder

Es folgen einige Beispiele, in denen mechanische Einwirkungen Grund für Lesefehler sein können:

Beispiel 1: Verstellter Schreib-/Lesekopf

Nach einigen Monaten Betriebszeit verschiebt sich der sogenannte Schreib-/Lesekopf in seiner Position. Das Magnetband liegt nicht mehr genau am Lesekopf an. Durch den zu großen Abstand werden nur noch schwache Signale gelesen, die als Folge eine Fehlermeldung erzeugen.

Dies kann im Laufe der Zeit zu einem echten Problem werden, wenn jeder Schneider-Besitzer seine Schreib-/Leseköpfe neu justiert! Dadurch, daß alle Schreib-/Leseköpfe unterschiedlich eingestellt sind, wird es über kurz oder lang Lesefehler beim Laden von Kassettensoftware geben, die vielleicht von einem anderen Schneider-Besitzer aufgenommen wurde. Solche Dinge sind heute z.B. beim Commodore 64 an der Tagesordnung.

Abhilfe läßt sich nur schaffen, wenn der Schreib-/Lesekopf richtig justiert wird.

Beispiel 2: Beschädigung des Bandmaterials

Auch hier liegen wieder mechanische Beschädigungen vor. Durch die Reibung am Lesekopf des Kassettenrekorders kommt es irgendwann zu Beschädigungen des Trägerbandes. Ein solcher Fehler tritt meist erst nach längerem Gebrauch einer Kassette auf. Oft ist es aber dann auch das letzte Mal, daß man mit dieser Kassette gearbeitet hat. Versuchen Sie in einem solchen Fall, diejenigen Programme zu retten, die sich noch lesen lassen, und werfen dann die defekte Kassette sofort in den Papierkorb.

Eine weitere Fehlerursache, mit der man leben muß, sind nagelneue Kassetten, die vom Bandmaterial her nicht einwandfrei sind. Die Magnetschicht des Bandes kann Stellen aufweisen, an denen kein magnetisches Material vorhanden ist. Diese Stellen sind mit dem Auge nicht zu erkennen; man nennt sie in der Fachsprache "Drop Outs".

Beispiel 3: Falsche Lagerung von Datenkassetten

Auch dieser Punkt sollte genau beachtet werden! Durch unsachgemäße Aufbewahrung von Kassetten ging schon so manches Programm verloren. Die jetzt geschilderte Situation kennt wahrscheinlich jeder:

Bis spät abends ist man noch am Programmieren und vergißt dabei die Zeit. Sieht man dann irgendwann auf die Uhr, wird anschließend der Tag recht schnell beendet. Da man müde ist und zum Aufräumen des Computerarbeitsplatzes keine rechte Lust mehr hat, bleibt alles liegen und stehen. Oft liegen in solchen Fällen Datenkassetten in der Nähe von Steckdosen, Trafos oder anderen stromführenden Kabeln. Auf magnetische Datenträger wirken diese Dinge aber zerstörend, d.h. es entsteht ein Datenverlust durch starke Magnetfelder, die durch die Netzspannung von 220 V Wechselstrom erzeugt werden.

Sollten also Ihre Datenträger eine Nacht lang in unmittelbarer Nähe einer Steckdose oder eines Trafos gelegen haben, können Sie davon ausgehen, daß es beim Laden von Programmen und Daten zu Lesefehlern kommt. Wenn Sie Ihre Arbeit am Computer beendet haben, legen Sie daher die Datenträger an einen Ort, an dem es zu solchen Pannen nicht kommen kann.

Außerdem empfehle ich Ihnen, zum Arbeitsende von allen Programmen Kopien anzufertigen, um auf Nummer Sicher zu gehen.

Beispiel 4: Bandsalat

Mit dieser Gefahr muß man bei der Verwendung von Kassetten leben. Wenn der Fall einmal eintritt, sollten Sie vorsichtig versuchen, ob sich die auf Band befindlichen Daten noch lesen lassen. Wenn ja, machen Sie sofort eine Datensicherung, damit die Programme gerettet werden. Ist dieses erfolgt, werfen Sie die Kassetten, bei denen der Bandsalat aufgetreten ist, weg.

Sollten sich die Programme nicht mehr laden lassen, haben Sie hoffentlich eine Kopie davon angefertigt. Ansonsten ist leider viel Programmierarbeit umsonst gewesen. Sollte also ein Fehler beim Laden vorkommen, erkennen Sie ihn an der Meldung: *Read Error*.

Manchmal können Sie sich damit helfen, daß Sie den gesamten Ladevorgang wiederholen. Wenn sich dann jedoch auch nach mehreren vergeblichen Versuchen noch nichts getan hat, müssen Sie sich mit dem Gedanken vertraut machen, daß dieses Programm verloren ist.

Ich rate Ihnen auf jeden Fall davon ab, bei einem auftretenden Lesefehler den Schreib-/Lesekopf mit einem Schraubenzieher zu verstellen und den Lesevorgang dann zu wiederholen. In einem solchen Fall kann es Ihnen passieren, daß sich Ihre anderen Programme nicht mehr laden lassen, weil die Position des Lesekopfes verstellt wurde. Ich würde nur dann zu dieser Methode greifen, wenn alles andere nichts mehr hilft und Sie gerade auf dieses eine Programm angewiesen sind.

6.6 Der Befehl CAT

Wie Sie bereits wissen, werden Programme mit einem Namen versehen und anschließend abgespeichert. Um sich einen Überblick über alle Programme zu verschaffen, die sich auf einer Kassette befinden, benötigt man den Befehl CAT. Dieser Befehl wird ohne Zusatz verwendet.

CAT kommt von dem Begriff CATalog; die Anweisung gibt eine Übersicht über alle auf der Kassette gespeicherten Programme auf dem Bildschirm aus. Dabei erscheint hinter dem Programm-Namen auch noch die Anzahl der Blöcke, die Programmart und die Buchstabenfolge "ok" für einwandfrei gespeicherte Programme. Die Hinweise für die Programmart sehen folgendermaßen aus:

Das Zeichen	**steht für**
$	ein normal gespeichertes BASIC-Programm
%	ein geschützt gespeichertes BASIC-Programm
*	ein als ASCII-Datei gespeichertes Programm
&	ein binär gespeichertes Programm

Wenn Sie beispielsweise fünf Programme auf Kassette gespeichert haben und diese mit CAT auflisten möchten, könnten folgende Meldungen erscheinen:

```
UHR          block 1     $     ok
TEXTVERARB   block 4     *     ok
SPIEL        block 5     &     ok
DATEI        block 9     %     ok
KALKULATION  block 8     %     ok
```

Dabei ist das Programm UHR ein normales BASIC-Programm, das Programm TEXTVERARB eine ASCII-Datei, das Programm SPIEL ein binär gespeichertes Programm und die beiden Programme DATEI und KALKULATION zwei geschützte Anwenderprogramme.

Wichtig ist noch die Tatsache, daß Programme im Arbeitsspeicher des Computers bei Verwendung von **CAT** nicht zerstört werden. Außerdem können Sie den **CAT**-Befehl in Ihre Programme übernehmen, falls dies gewünscht wird. Eine Programmzeile könnte dann beispielsweise so aussehen: `60 CAT`.

Insgesamt gesehen ist der Befehl CAT für die Programmierarbeit sehr vorteilhaft.

6.7 Arbeiten mit dem Diskettenlaufwerk

Wer einmal mit der Kassette als Speichermedium gearbeitet hat, weiß um die Unzulänglichkeiten derselben. Mit der Diskette fallen alle Nachteile der Kassette weg. Fairerweise sollte man auch den Preis einer Diskette erwähnen, der im Vergleich zur Kassette recht hoch ist. Dies hängt wohl auch damit zusammen, daß es zur Zeit nur einen einzigen Hersteller gibt, der Disketten der Größe 3-Zoll herstellt. Sie bezahlen heute für eine Diskette dieser Größe fast 10mal mehr, als für eine Kassette der Länge C10.

Fraglich ist auch, ob sich das Format von 3-Zoll Diskettendurchmesser letztendlich auf dem Markt durchsetzen wird, da fast alle Computerhersteller das etwas größere Format von 3.5-Zoll verwenden. Kritisch zu sehen ist auch die Werbung, daß unter dem Betriebssystem CP/M schon Tausende von Anwenderprogrammen zur Verfügung stehen. Im Prinzip stimmt die Aussage zwar, nur gibt es diese Programme zur Zeit noch auf den normalen 5.25-Zoll-Disketten, die im Bereich Home- und Personal-Computer gängig sind.

Vergleichen Sie einmal die Preise: Für die leeren 3-Zoll- und 5.25-Zoll-Disketten bekommen Sie heute für den Preis einer 3-Zoll-Diskette schon drei Stück der normalen Größe von 5.25-Zoll. Der Trend geht zwar zu immer kleineren Disketten hin, aber bei einem solchen Preis-/Leistungsverhältnis kann ich mir nicht recht vorstellen, daß der Markt dies ohne weiteres mitmacht. Das Käuferverhalten ist außerdem in den letzten Jahren wesentlich kritischer geworden, da das Angebot heute umfangreicher ist als jemals zuvor. So ist man beispielsweise nicht mehr unbedingt auf Zusatzgeräte angewiesen, die direkt vom Hersteller kommen, sondern kann sich auch auf dem freien Markt umsehen, ob es vielleicht nicht etwas besseres gibt.

Diese Tatsache wird die Hersteller vermutlich dazu veranlassen, sich dem Markt anzupassen und ihre Produktpalette den Käuferwünschen entsprechend anzugleichen. Wenn dann auch noch der Preis stimmt, bleibt nichts mehr zu wünschen übrig.

Nun wieder zurück zu unserer 3-Zoll-Diskette. Von ihrer Konstruktion her ist diese Diskette etwas Neues, weil sie in einem stabilen Plastikgehäuse eingebaut und mit einer kleinen Platte vor Staub geschützt ist. Ebenfalls neu gegenüber den normalen 5.25-Zoll Disketten ist die Art des Schreibschutzes. Sie kennen es wahrscheinlich von Ihren Musikkassetten her, daß man an der Unterseite zwei Laschen herausbrechen muß, damit mit einer bespielten Kassette keine weiteren Aufnahmen gemacht werden können und so ein ungewolltes Überspielen einer Aufnahme verhindert wird.

Nach demselben Prinzip funktioniert auch eine Diskette. Bei einer 5.25-Zoll-Diskette wird durch Überkleben einer Kerbe verhindert, daß auf diese Diskette weitere Daten gespeichert werden. Möchten Sie später doch noch ein Programm darauf speichern, müssen Sie den Aufkleber wieder abreißen, das neue Programm speichern und schließlich die Kerbe wieder überkleben. Hat man einen solchen Vorgang mehrmals wiederholt, ist das Umfeld der Kerbe allerdings vollkommen verklebt.

Anders sieht es bei den 3-Zoll-Disketten aus: Erstens können Sie die Disketten von beiden Seiten benutzen, zweitens sitzt an jeder Seite ein Schieber, der als Schreibschutz dient. Diesen Schieber können Sie bequem mit einem Fingernagel in die gewünschte Position bringen. So läßt sich ein Schreibschutz schnell anbringen und ebenso schnell wieder entfernen. Einfacher geht es wirklich nicht mehr! Insgesamt wirkt also die 3-Zoll-Diskette robuster als die normalen Disketten der Größe 5.25-Zoll.

Wenn Sie Ihr Diskettenlaufwerk aufstellen, achten sie bitte auf einen ausreichenden Abstand zum Monitor, da magnetische Felder auch hier die Datenübertragung vom Computer zur Diskettenstation erheblich stören können.

Wie Sie mit einer Diskette arbeiten, werden wir im folgenden Abschnitt erörtern.

6.7.1 Speichern von Programmen auf Diskette

Den Befehl SAVE zur Datenspeicherung kennen Sie ja bereits. Bei der Verwendung eines Diskettenlaufwerks gibt es einige Besonderheiten zu beachten:

Sie müssen darauf achten, daß sich auch wirklich eine Diskette im Laufwerk befindet, sonst meldet der Rechner:

```
Drive A: disc missing
Retry, Ignore or Cancel?
```

Diese Meldung bedeutet, daß im Laufwerk A keine Diskette liegt und der Computer fragt sie nun, ob wiederholt, ignoriert oder abgebrochen werden soll.

Ihre Programmnamen dürfen jetzt nur noch eine Länge von 8 Zeichen haben, da das DOS (Disketten-Operating-System) noch einige Zeichen anhängt. Diese Zeichen sind:

.	meistens ein als ASCII-Datei abgespeichertes Programm
.BAS	ein abgespeichertes BASIC-Programm
.BAK	ein abgespeichertes BASIC-Programm, das aber unter diesem Programm-Namen bereits existiert.
.BIN	ein binär gespeichertes Programm, z.B. Bildschirminhalt
.COM	alle Befehlsdateien auf der CP/M-Diskette

6.7.2 Laden von Programmen von der Diskette

Dazu können Sie ebenfalls den schon bekannten Befehl **LOAD** verwenden. Allerdings sollten Sie hier den Namen des Programms angeben, das Sie suchen, da es sonst zu Schwierigkeiten kommt.

Wenn Sie ein Programm direkt von Diskette starten möchten, geben Sie einfach ein:

```
RUN"Name des Programms
```

Sicherlich wird Ihnen auffallen, mit welcher Geschwindigkeit die Programme geladen werden. Das ist der Vorteil einer Diskette gegenüber einer Kassette.

6.7.3 Der Befehl CAT bei Verwendung von Disketten

Bei jedem Programm oder bei jeder Sammlung von Daten legt sich das DOS automatisch ein Inhaltsverzeichnis an. Darin befinden sich der Name des Programms, die Art der Speicherung und die Größe des Programms in Kbyte.

Als weitere Besonderheit werden alle Programme alphabetisch auf dem Bildschirm gelistet und somit eine gute Übersichtlichkeit garantiert.

Anhand des folgenden Beispiels können Sie sehen, wie ein solches Verzeichnis aussehen kann. Sie geben dazu den Befehl **CAT** ein und drücken die Taste <ENTER>. Nun erscheint auf dem Bildschirm:

```
CAT
Drive A:              user 0
AMSDOS     .COM       1K      GRAFIK      .BAS     2K
BILD       .BIN       17K     UHR         .BAS     1K
BILD2      .BIN       17K     WECKER      .BAS     2K
DATEI01    .BAS       10K     ZEICHNEN    .BAS     12K
DATEI02    .BAS       12K
EINGABE    .          2K
EINGABE1   .BAS       1K
```

In dem eben gezeigten Beispiel finden Sie verschiedene Arten von Programmen. Das erste ist eine Befehlsdatei, gefolgt von zwei Binärdateien und anschließend einigen BASIC-Programmen. Steht nichts hinter dem Namen, so handelt es sich um ein Programm, das als ASCII-Datei gespeichert wurde.

Wichtig bei dem Befehl **CAT** ist, daß Ihre Programme, die sich gerade im Speicher befinden, bei diesem Aufruf nicht verloren gehen. Es gibt noch einen anderen Befehl für das Inhaltsverzeichnis. Er lautet **DIR** und zeigt die Reihenfolge der Speicherung an. Dieser Befehl wird in erster Linie unter dem Betriebssystem CP/M eingesetzt.

7 Nützliche Programmierhilfen

In diesem Kapitel werde ich Ihnen einige leistungsfähige Befehle vorstellen, mit deren Hilfe Sie Ihre Arbeit am Schneider erheblich komfortabler und effektiver gestalten können. Doch zunächst möchte ich diese Befehle kurz vorstellen und danach speziell auf jeden einzelnen eingehen.

Befehl	**Funktion/Aufgabe**
AUTO	automatische Zeilennumerierung
EDIT	holt eine zu ändernde Programmzeile unten auf den Bildschirm
FRE(X)	gibt als PRINT FRE(X) den noch freien Platz an Arbeitsspeicher an
KEY	legt eine Befehlsfolge oder eine Zeichenkette auf eine Taste, die dann als Funktionstaste arbeitet
RENUM	numeriert ein Programm um und ergibt einen neuen Zeilennummernbereich
STOP	unterbricht ein Programm in der angegebenen Zeilennummer
TRON	(TRACE ON) - automatische Ausgabe der aktuellen Zeilennummer, die gerade berarbeitet wird auf dem Bildschirm
TROFF	(TRACE OFF) - die vorherige Funktion wieder außer Kraft setzen

Viele dieser Befehle sind bei einigen Konkurrenzmodellen, z.B. bei dem Commodore 64, nicht vorhanden und werden im Rahmen sogenannter "BASIC-Erweiterungen" angeboten. Der Nachteil dabei ist, daß diese Befehlserweiterungen meist vor Arbeitsbeginn in den Computer geladen werden müssen, wenn es sich nicht um Steckmodule handelt. Sie sehen, der Schneider ist hier schon recht gut bestückt.

Als erstes möchte ich Ihnen nun den Befehl AUTO vorstellen, der die Vergabe von Zeilennummern während der Programmierung übernimmt.

7.1 Der Befehl AUTO

Mit diesem Befehl haben Sie die Möglichkeit, sich die Arbeit der Zeilennumerierung zu ersparen, da der Rechner diese Aufgabe für Sie durchführt. Natürlich müssen Sie dem Schneider vorher mitteilen, ab wo er mit seiner Arbeit beginnen und mit welcher Schrittweite gearbeitet werden soll. Der Befehl AUTO hat folgendes Format:

```
AUTO Anfangszeilennummer, Schrittweite
```

Möglich sind auch folgende Versionen:

```
AUTO
```

oder

```
AUTO Anfangszeilennummer
```

oder

```
AUTO ,Schrittweite
```

Normalerweise beginnt die Numerierung bei dem Wert 10, aber auch `AUTO 1,1` funktioniert. Ich empfehle Ihnen, bei Zeilennummer 10 zu beginnen und mit einer Schrittweite von 10 fortzufahren. Damit haben Sie die Möglichkeit, später noch Zeilennummern einzufügen. Mit einer Schrittweite von 100 oder größer wird das ganze Programm schnell unübersichtlich; außerdem kosten Zeilennummern ja auch Speicherplatz. Beispielsweise ergibt AUTO die Zeilennummern 10, 20, 30 usw. `AUTO 100` beginnt denselben Vorgang ab Zeile 100. Also 100, 110, 120 usw. `AUTO ,200` beginnt mit der Arbeit bei 10 und addiert dann jedesmal 200 dazu. Dies ergibt die Folge 10, 210, 410, 610 usw.

Achtung! Zeilen, die bereits belegt sind, werden mit einem Stern gekennzeichnet, zum Beispiel so: 10*

In diese Zeile sollten Sie nichts mehr schreiben, es sei denn, Sie möchten diese Zeilennummer überschreiben.

7.2 Der Befehl EDIT

Mit dem EDIT-Befehl können Sie längere Programmzeilen schneller auf dem unteren Bildschirmbereich zur Änderung zur Verfügung stellen, als es mit der umständlichen <COPY>-Taste möglich ist. Sie brauchen nur

eingeben: `EDIT 10` und die Zeilennummer steht zum Verändern zur Verfügung. Durch Bestätigung mit <ENTER> wird sie als neue Zeile 10 im Arbeitsspeicher des Rechners abgelegt. Diese Möglichkeit sollten Sie meiner Ansicht nach möglichst häufig nutzen und eventuell den Befehl `EDIT` auf eine Taste legen, die Sie dann als Funktionstaste benutzen können.

7.3 Der Befehl FRE(X)

Anfangs brauchen Sie sich über freien Speicherplatz wohl keine allzu großen Sorgen machen, denn dieser ist mit rund 43.5 Kbyte schon recht anständig bemessen. Auch wenn der Trend in dieser Leistungsklasse zu Systemen mit 128 Kbyte geht, müssen Sie erst einmal mit einem selbstgeschriebenen Programm die Grenze von rund 40 Kbyte erreichen, um zu ermessen, um welche Größe es sich dabei überhaupt handelt. Für viele Problemlösungen ist eine solche Speicherplatzgröße voll ausreichend, auch wenn mancher darüber die Stirn runzelt.

Gerade der begrenzte Speicherplatz zwingt den Benutzer jedoch zu Überlegungen, wie ein größeres Programm platzsparend aufgebaut werden soll. Bei einer Speicherplatzgröße von 128 Kbyte oder mehr wird wohl jeder verschwenderischer mit dem Platz umgehen als bei einem kleineren Bereich. Aber gerade der große Speicherplatz verleitet zur "unsauberen Programmierung".

Bei einem Datenverwaltungsprogramm ist es oft recht interessant zu wissen, wieviel Speicherplatz für die Daten noch zur Verfügung steht. Mit `PRINT FRE(X)` bekommt man den freien Platz an Arbeitsspeicher in Byte auf dem Bildschirm angezeigt. Eine Besonderheit gibt es bei dieser Funktion, die hier noch erwähnt werden sollte. Das Zeichen in Klammern kann ein beliebiges Zeichen sein, denn es handelt sich dabei um ein sogenanntes Scheinargument. Ein Scheinargument ist ein Zeichen, das der Rechner zwar braucht, dessen Wert bei der Verarbeitung jedoch keine Rolle spielt.

Das folgende Programm berechnet den freien Arbeitsspeicher, der bei jedem Programmdurchlauf kleiner wird.

```
10    MODE 1
20    CLS
30    FOR I=1000 TO 8000 STEP 200
40    DIM A(I)
50    LOCATE 1,1
```

```
60    PRINT"Aktueller Schleifenwert:    ";I
70    PRINT"Freier Speicherplatz:       ";FRE(X);" Byte "
80    ERASE A
90    NEXT I
```

Das Programm durchläuft die Werte von 1000 bis 8000 jeweils in Schritten von 200. Mit dem aktuellen Wert der Schleife wird dann auch dimensioniert und der freie Speicherplatz in Byte angezeigt. Weil nicht innerhalb eines Programms mehrmals die gleiche Variable dimensioniert werden kann, wird mit dem Befehl ERASE A diese Variable wieder gelöscht und im näohsten Durchlauf der Schleife neu dimensioniert.

Wenn Sie den freien Speicherplatz in Kbyte sehen möchten, brauchen Sie nur zu schreiben

```
PRINT FRE(X)/1000
```

und die Ausgabe erscheint in Kbyte auf dem Bildschirm. Empfehlenswert ist auch hier wieder die Belegung einer bestimmten Taste mit diesem Befehl.

7.4 Der Befehl KEY

Beim Programmieren ist es oft recht lästig, immer wieder lange Befehlsfolgen Zeichen für Zeichen eingeben zu müssen. Besser wäre es, wenn man, wie bei PRINT das Fragezeichen, eine Taste drückt und der fertige Befehl erscheint auf dem Bildschirm. Beim Schneider ist ein solcher Befehl vorhanden. Er lautet KEY und hat folgendes Darstellungsformat:

```
KEY Tastennummer, Befehlsfolge
```

oder

```
KEY Tastennummer, Zeichenfolge
```

Programmbeispiel:

```
KEY 128,"LIST"
```

Dieser Befehl bedeutet, daß auf der Zehnertastatur die Taste <0> mit dem Befehl LIST belegt wird. Drücken Sie anschließend auf diese Taste, erscheint nicht mehr die 0, sondern der Befehl LIST. Unvorteilhaft ist dabei, daß danach immer die ENTER-Taste gedrückt werden muß. Aber auch hier läßt sich auf einfache Art Abhilfe schaffen. Der ASCII-Code

für ENTER beträgt 13. Wird dieser Wert an den Befehl LIST angehängt, simuliert man den Druck auf die <ENTER>-Taste und der Befehl wird sofort ausgeführt. Konkret sieht die Anweisung dann so aus:

```
KEY 128,"LIST"+CHR$(13)
```

Nun wird der Befehl sofort ausgeführt, wenn auf die entsprechende Taste, in diesem Fall die Taste <0> gedrückt wird. Sinnvollerweise belegt man den Zehnerblock mit Befehlen, denn diesen kann man am ehesten entbehren. So entstehen die sogenannten Funktionstasten, die bei Druck auf eine bestimmte Taste etwas vorbestimmtes ausführen.

Neben Befehlen können auch Texte auf die Funktionstasten (Zehnerblock) gelegt werden, um sich die Arbeit bei ständig wiederkehrenden Texten zu erleichtern. Denkbar ist hier der Einsatz in einer Adressenverwaltung, wo es bei der Anrede nur Herr, Frau oder Fa. gibt. Werden diese drei Texte auf bestimmte Tasten gelegt, erspart man sich eine Menge Tipparbeit. Beispielsweise könnte eine Belegung so aussehen:

```
KEY 128,"Herr":KEY 129,"Frau":KEY 130,"Fa."
```

Zum Schluß noch ein kleines Programm, das den Zehnerblock belegt.

```
10    MODE 1
20    CLS
30    KEY 128,"LIST -200"+CHR$(13)
40    KEY 129,"LIST 200-"+CHR$(13)
50    KEY 130,"RENUM"
60    KEY 131,"RUN"+CHR$(13)
70    KEY 132,"MODE"
80    KEY 133,"CLS"+CHR$(13)
90    KEY 134,"CALL&BC02"+CHR$(13)+"INK 1,24"+CHR$(13)+"PEN 1"+CHR$(13)+
      "CLS"+CHR$(13)
100   KEY 135,"INPUT"+CHR$(34)
110   KEY 136,"SOUND"
120   KEY 137,"BORDER"
130   KEY 138,"CAT"+CHR$(13)
140   KEY 139,"AUTO"
150   KEY 140,"EDIT"
```

Sie finden hier die wichtigsten Befehle, die beim Programmieren immer wieder gebraucht werden. Zeile 90 ist die "Notbremse" für die Farbeinstellung, wenn mal "nichts mehr geht". Die Befehle erreichen Sie wie folgt:

Taste	Befehl	Taste	Befehl
0	LIST-200	6	Farbe einstellen
1	LIST 200-	7	INPUT
2	RENUM	8	SOUND
3	RUN	9	BORDER
4	MODE	PUNKT	CAT
5	CLS	ENTER	AUTO
		CTRL+ENTER	EDIT

7.5 Der Befehl RENUM

Oft stellt man während des Programmierens fest, daß man noch einige Zeilen zusätzlich zwischen bereits vorhandene Zeilennummern einsetzen möchte. Falls aber nicht mehr genügend Platz vorhanden ist, was tun?

Hier bietet sich der Befehl RENUM an, der Teile des Programms oder sogar das gesamte Programm umnumeriert. Er hat folgendes Format:

```
RENUM erste Zeilennummer des neuen Programms,
erste Zeilennummer des alten Programms,
gewünschte Schrittweite
```

Wird nur RENUM eingegeben, beginnt der Vorgang der Neunumerierung ab Zeile 10 mit einer Schrittweite von 10. Die folgenden Beispiele sollen einen Überblick über mögliche Kombinationen geben:

RENUM numeriert ein Programm ab Zeile 10 mit der Schrittweite von 10 um.

RENUM `100,10,100` die erste Zeilennummer des neuen Programms beginnt mit 100 und das Programm bekommt die Schrittweite 100.

RENUM `100` das Programm wird umnumeriert, wobei die Zeilennummern später bei 100 anfangen und eine Schrittweite von 10 haben.

Umnumeriert werden dabei nicht nur die Zeilennummern vorne, sondern auch etwaige Sprungbefehle werden korrekt behandelt und bearbeitet.

Nachteilig ist, daß sich Teilbereiche nicht umnumerieren lassen, wie etwa der Bereich 4000-5000, wenn über 5000 noch weitere Zeilennummern folgen. Beachtet man dies jedoch, möchte man bald auf diesen Befehl nicht mehr verzichten.

7.6 Der Befehl STOP

Beim Testen von Programmen ist es nützlich, wenn man einzelne Teilbereiche des Programms bis zu einer bestimmten Stelle laufen lassen kann, um diesen Bereich dann auf Fehler zu überprüfen. Beispielsweise kann man sich an einer Übergangsstelle zum nächsten Programmteil die Variablen ausgeben lassen, bevor diese im nächsten Teil weiter verarbeitet werden.

Um ein Programm zu unterbrechen, benutzt man den Befehl STOP. Kommt das Programm dann an die Zeilennummer mit dem STOP, wird es unterbrochen mit der Meldung *"Break in ..."*. Dabei bleiben alle aktuellen Werte in den Variablen erhalten und können auf Wunsch abgerufen werden. Ein Beispiel dazu:

```
10    MODE 1
20    CLS
30    FOR I=1 TO 10
40    A(I)=I*I
50    NEXT I
60    STOP
70    FOR I=1 TO 10
80    PRINT A(I)
90    NEXT I
```

Das Programm wird mit der Meldung *"Break in 60"* unterbrochen. Mit dem Befehl CONT für Continue fährt das Programm in seiner Ausführung fort. Dabei hat der Programmierer die Möglichkeit, den ersten Programmteil zu testen und dann mit CONT zum zweiten Programmteil überzugehen.

7.7 Die Befehle TRON und TROFF

Bei Programmierfehlern innerhalb eines Programms ist es sehr hilfreich zu sehen, wo der Rechner sich gerade befindet und was er dort macht. In solchen Fällen empfehle ich Ihnen ein Listing des entsprechenden Programms neben den Schneider zu legen und jede Zeile mit dem Listing zu vergleichen, das sich im Arbeitsspeicher des Programms befindet. Schreibfehler, also Fehler, die einen *"Syntax error"* erzeugen, können Sie auf den ersten Blick erkennen.

Wie sieht es aber mit logischen Fehlern aus? Zum Beispiel steht in einer Programmierzeile: GOTO 100, was vom Programm her gesehen falsch ist. Die Zeile wird jedoch korrekt ausgeführt, da der Befehl GOTO 100

richtig geschrieben wurde. Solche Fehler sind reine Überlegungsfehler, die die meisten Schwierigkeiten bei der Fehlersuche verursachen, da sie auf den ersten Blick nicht zu erkennen sind.

Es gibt jedoch die Möglichkeit, die Zeilennummern, die gerade bearbeitet werden, auf dem Bildschirm sichtbar zu machen. Dies geschieht mit Hilfe der Funktion TRACE. Mit dem Befehl TRON, also TRACE ON, wird diese Hilfsfunktion eingeschaltet. Die aktuellen Zeilennummern stehen dann in eckigen Klammern vor dem Zeilenanfang auf dem Bildschirm. TROFF (TRACE OFF) schaltet die Funktion wieder aus.

Oft ist der Befehl TRON die einzige Möglichkeit, hartnäckigen Fehlern auf die Schliche zu kommen. Üben Sie sich deshalb im Umgang mit diesen beiden Befehlen, denn sie können für Sie noch recht hilfreich sein.

8 Positionierte Datenausgabe

Bisher haben wir nur darauf geachtet, "was" auf dem Bildschirm ausgegeben wurde und nicht "wie" es ausgegeben wurde. In diesem Kapitel geht es nun darum, wie Tabellen, Listen und andere formatierte Ausgaben auf Bildschirm und Drucker ausgegeben werden. Dazu stellt der Rechner einige leistungsfähige Befehle zur Verfügung:

TAB	Tabulator innerhalb einer Zeile
SPC	Abstand (Leerzeichen) zwischen 2 Ausgaben innerhalb einer Zeile
LOCATE	Positioniert den Textcursor an einer beliebigen Stelle
PRINT USING	Formatiert Texte und Zahlen

Wenden wir uns zunächst dem einfachsten Befehl zu.

8.1 Der Befehl TAB

Die Funktion von TAB kennen Sie vielleicht von der Schreibmaschine her; man kann damit innerhalb einer Bildschirmzeile eine beliebige Schreibposition anfahren und dort etwas ausgeben. Dazu gleich ein kleines Beispiel:

```
PRINT TAB(10);"Position 10";TAB(35);"Pos. 35"
```

Bitte geben Sie diesen PRINT-Befehl im Direktmodus ein und drücken Sie die <ENTER>-Taste. Ab der 10. und der 35. Position wird dann der kurze Text ausgegeben. Beim Befehl TAB steht die Position immer in Klammern, danach kommt die Angabe, was ausgegeben werden soll.

Zur Vertiefung fügen wir noch ein weiteres Beispiel an, bei dem drei Vor- und Nachnamen nebeneinander auf dem Bildschirm stehen sollen. Das Programm dazu sieht so aus:

```
10    PRINT TAB(5);"Werner";TAB(20);"Meyer"
20    PRINT TAB(5);"Thomas";TAB(20);"Wernecke"
30    PRINT TAB(5);"Heike";TAB(20);"Sudbrock"
```

Wie Sie sehen, kommen Sie mit dem TAB-Befehl an jede beliebige Position einer Zeile und können dort etwas ausgeben. Die Position, auf der etwas ausgegeben werden soll, ist dabei eine ganzzahlige Angabe.

Bitte beachten Sie nun einmal folgende Aufgabenstellung: Innerhalb einer Zeile sollen zwei verschiedene Positionen mit TAB angesteuert werden. Auf Position 10 soll das Wort "Test" beginnen und auf Position 13 das Wort "Versuch". Das Wort "Test" hat insgesamt 4 Buchstaben, was bedeutet, daß die Schreibpositionen 10, 11, 12, 13 belegt sind. Da aber im nächsten Schritt gesagt wird, daß auf Position 13 ein neues Wort anfangen soll, wird es Probleme geben, da auf der 13. Position noch der letzte Buchstabe von "Test" steht.

In einem solchen Fall, d.h. wenn eine Schreibposition bereits belegt ist, setzt der Computer den zweiten Ausdruck in die folgende Zeile. Sie können sich selbst davon überzeugen, wenn Sie diese beiden Zeilen eingeben:

```
10    PRINT TAB(10);"Test";
20    PRINT TAB(13);"Versuch"
```

Wird Zeile 20 so geändert, daß es zu keiner Überschneidung kommt, läuft das Programm wieder einwandfrei ab.

```
20    PRINT TAB(14);"Versuch"
```

Beispiel: Pfeil auf dem Bildschirm ausgeben

Das nächste Beispiel druckt Ihnen einen großen Pfeil aus, der aus Sternchen besteht. Dabei arbeitet eine Schleife mit der TAB-Funktion zusammen, d.h. bei jedem Schleifendurchlauf wird die Position mit TAB verändert.

Das Programm:

```
10    MODE 1
20    CLS
30    FOR I=10 TO 19
40    PRINT TAB(I);"******"
50    NEXT I
60    PRINT STRING$(25,"*")
70    PRINT STRING$(28,"*")
80    PRINT STRING$(25,"*")
90    FOR I=19 TO 10 STEP -1
100   PRINT TAB(I);"******"
110   NEXT I
120   PRINT
130   PRINT "Ende mit  zweimal ESC-Taste......"
140   GOTO 140
```

Programmbeschreibung:

Wichtig sind die Zeilen 30 bis 50 und 90 bis 110. Hier steht der Befehl TAB innerhalb einer Schleife und sorgt dafür, daß sich die Position ständig verändert. Die Sterne in den Zeilen 40 und 100 sind für die Breite des Pfeils zuständig. Die Zeilen 60 bis 70 stellen die Mitte des Pfeils dar, wobei Zeile 70 mit 28 Sternchen die Spitze des Pfeils darstellt.

Sie können das Programm beliebig verändern, und sollten dabei insbesondere mit dem TAB-Befehl ein wenig experimentieren.

8.2 Der Befehl SPC

Dieser Befehl arbeitet ähnlich wie TAB, bezieht sich aber nicht als Ausgangsposition auf den linken Zeilenrand, sondern auf die Position des letzten Ausdrucks in dieser Zeile.

Das nachfolgende Beispiel soll dies verdeutlichen.

```
10    CLS
20    PRINT TAB(10);"*";TAB(20);"*"
30    PRINT SPC(10);"*";SPC(20);"*"
```

Starten Sie das Programm mit RUN und sehen Sie sich das Ergebnis an.

Das Programm setzt in Zeile 20 auf die Positionen 10 und 20 je einen Stern. Obwohl Zeile 30 Zeile 20 sehr ähnlich ist, ist das Resultat doch vollkommen unterschiedlich. Hier stehen zunächst einmal 10 Leerzeichen und auf der 11. Position folgt ein Sternchen. Im zweiten Schritt der Zeile 30 wird zwischen dem ersten und dem zweiten Sternchen ein Abstand von insgesamt 20 Leerzeichen gelassen. Das letzte Sternchen erscheint also auf Position 32.

Sie können daran deutlich den Unterschied zwischen dem Befehl TAB und dem Befehl SPC erkennen. Beide Befehle positionieren beliebige Daten, arbeiten aber unterschiedlich.

TAB gibt immer die Stelle an, auf der die nächste Ausgabe erscheinen soll und bezieht sich dabei auf die Ausgangsposition am Anfang der jeweiligen Zeile.

SPC erzeugt Zwischenräume zwischen zwei verschiedenen Ausdrücken auf dem Bildschirm und beginnt dabei mit der Zählung immer nach dem letzten Zeichen des letzten Ausdrucks.

8.3 Der Befehl LOCATE

Während die beiden eben vorgestellten Positionierungsbefehle nur innerhalb einer Bildschirmzeile arbeiten, kann man mit LOCATE auf jeden Punkt des Bildschirms schreiben. Der Befehl LOCATE hat dabei folgendes Darstellungsformat:

LOCATE Position innerhalb einer Zeile, Zeile

Beispiel: LOCATE 5,10 setzt den Textcursor auf Position 5 in der Zeile 10. Mit einem nachfolgenden PRINT-Befehl kann dort etwas ausgegeben werden.

Leider stört es mich immer, daß es nicht möglich ist, die Angabe nicht umgekehrt einzugeben. Den Befehl LOCATE gibt es auch in anderen BASIC-Versionen, so z.B. beim IBM Personal Computer. Hier steht aber die Zeilennummer an erster Stelle und die Position innerhalb dieser Zeile an zweiter Stelle.

Im folgenden Beispiel wird wieder mit einer Schleife und LOCATE positioniert.

```
10   CLS
20   A$="Versuchsprogramm"
30   FOR I=1 TO 20
40   LOCATE I,I
50   PRINT A$
60   NEXT I
```

In Zeile 40 positioniert das Programm jedesmal den Textcursor neu und gibt den Text in A$ genau auf dieser Position aus.

LOCATE findet sehr oft Verwendung in der Programmierung von Eingabemasken zur Erfassung von Daten. Ein entsprechendes Beispiel finden Sie im Kapitel Standardgrafikzeichen.

8.4 Der Befehl PRINT USING

Der wohl umfangreichste Positionierungsbefehl ist PRINT USING. Umfangreich deshalb, weil damit vielfältige Positionierungsmöglichkeiten für Texte und Zahlen zur Verfügung stehen. Der Befehl PRINT USING wird folgendermaßen eingesetzt:

```
PRINT USING "Format für Texte oder Zahlen",
Liste von Variablen
```

Folgende Möglichkeiten stehen dabei zur Verfügung:

Für Texte:

!	nur das erste Zeichen einer Zeichenkette wird hier ausgegeben.
&	Alle Zeichen werden ausgegeben, unabhängig von ihrer Länge.
\ \	Diese Schrägstriche arbeiten als Platzhalter für einen Text. Ist der Text länger als der Platzhalter, werden nur die ersten Zeichen beachtet, die in dieses Format passen.

Für Zahlen:

#	Ergibt eine Stelle für eine einzige Zahl.
####	Ergibt vier Stellen für eine ganzzahlige Zahl.
##.#	Ergibt eine Zahl mit zwei Stellen vor und einer Stelle nach dem Komma.
+	Gibt das Vorzeichen aus.
-	Gibt das Vorzeichen aus.
,	Alle drei Vorkommastellen wird eine Zahlenkolonne durch Kommas getrennt, z.B. 1,000,000 für 1000000.

Neben den hier aufgeführten Möglichkeiten für numerische Daten gibt es noch einige andere Formatierungszeichen wie beispielsweise den Stern (*) oder das Dollarzeichen ($). Auf diese Zeichen soll aber nicht näher eingegangen werden, da sie in der Praxis weniger häufig Verwendung finden. Nachfolgend sind zum Befehl PRINT USING einige Beispiele aufgeführt, die Sie zum besseren Verständnis einmal durchgehen sollten.

Beispiel 1: Formatieren von Texten

Aufgabe:

Es sollen mehrere Begriffe mit einer Länge von maximal... Zeichen ausgegeben werden. Als Begriffe dienen hier die Zahlen von eins bis zehn.

Eingabe:

Wird vom Programm selbst übernommen

Verarbeitung:

Daten über eine READ-Anweisung einlesen
Mit PRINT USING das Format festlegen
Alle Daten nacheinander ausgeben

Ausgabe:

Unformatierte Texte und anschließend dahinter die mit PRINT USING formatierten Texte

Das Programm:

```
10    CLS
20    DATA eins, zwei, drei, vier, fünf, sechs
30    DATA sieben, acht, neun, zehn
40    FOR I=1 TO 10
50    READ ZAHL$
60    PRINT"Normal:";TAB(15);ZAHL$
70    PRINT"Formatiert:";
80    PRINT TAB(15);USING"\  \";ZAHL$
90    PRINT
100   NEXT
```

Programmbeschreibung:

Die Zeile 10 löscht den Bildschirm. In den folgenden Zeilen werden die DATA-Angaben abgelegt.Anschließend wird mit einer Schleife nach und nach jeder Begriff eingelesen. Zuerst wird in Zeile 60 der normale und unbehandelte Text ausgegeben. In Zeile 80 folgt das Kernstück des Programms. Hier wird ein Text mit nur maximal 4 Zeichen zugelassen. Zwischen den Zeichen "\ \" stehen insgesamt 4 Zeichen, wobei die Zeichen \ mitgezählt werden!!

Beispiel 2: Formatieren von Zahlen

Aufgabe:

In diesem Kapitel soll eine Reihe von Zufallszahlen formatiert werden. Dabei kommt auch die Möglichkeit der Rundung zum Tragen, denn mit PRINT USING können auch Daten gerundet werden.

Eingabe:

Übernimmt in diesem Programm der Zufallsgenerator

Verarbeitung:

Zufallszahlen erzeugen
Zufallszahlen runden
Zufallszahlen ausgeben

Ausgabe:

Laufende Nr.
Zufallszahl ungerundet
Zufallszahl gerundet und formatiert

Das Programm:

```
10    MODE 1
20    INK 0,0
30    BORDER 0
40    CLS
50    FOR I=1 TO 14
60    PRINT I;
70    Z=RND(1)*1000
80    PRINT TAB(5);Z;
90    PRINT TAB(20);USING"###.##";Z;
100   PRINT TAB(28);USING"###.#";Z;
110   PRINT TAB(36);USING"###";Z;
120   PRINT
130   NEXT I
```

Programmbeschreibung:

In Zeile 70 wird die Zufallszahl erzeugt, die später bearbeitet wird. In den Zeilen 90 bis 110 wird entsprechend gerundet.

9 Arbeiten mit Tabellen und Feldern

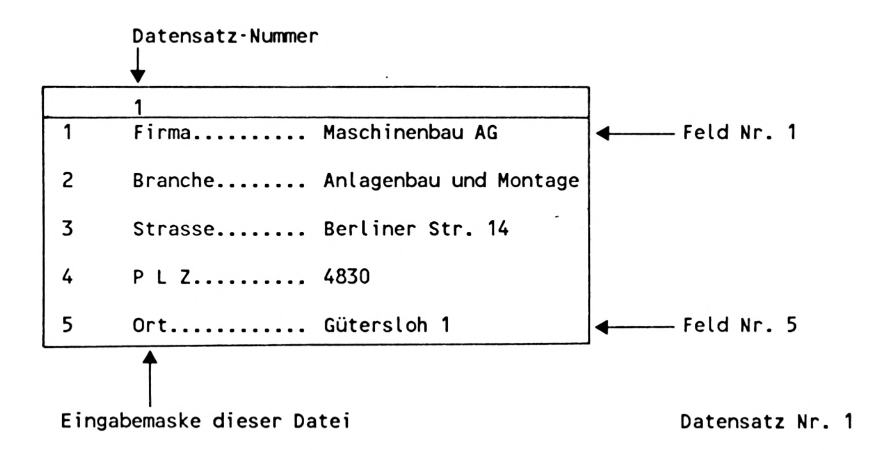
In diesem Kapitel geht es um die Verarbeitung von größeren Datenmengen, die alle in einem bestimmten Zusammenhang stehen. Es kann sich dabei beispielsweise um die Verwaltung von Adressen handeln, wobei sich jede Adresse aus bestimmten Teilen zusammensetzt. Die für diese Aufgabe verwendeten Fachbegriffe werde ich Ihnen kurz am Beispiel demonstrieren.

9.1 Grundbegriffe mit Beispielen

Am besten kann man die Bedeutung der Fachbegriffe an einer Adresse erkennen. Doch zunächst die Erläuterung einiger Begriffe:

Datenfeld	Teil eines Datensatzes, kleinste Einheit
Datensatz	Mehrere Datenfelder ergeben einen Datensatz
Datensatznr.	Nummer eines Datensatzes
Datenfeldnr.	Nummer eines Feldes innerhalb eines Datensatzes
Datei	Zusammenfassung mehrerer Datensätze, die alle in einem bestimmten Zusammenhang stehen
Datenbank	Ansammlung von verschiedenen Dateien, auf die beliebig zugegriffen werden kann und die gleichzeitig miteinander kombiniert werden können.

Wenn heute beispielsweise Datenbanken für Kleincomputer angeboten werden, dann handelt es sich immer um Dateien, weil Computer der untersten Preisklasse eine Verarbeitung von großen Datenbeständen mit gleichzeitigem Zugriff technisch gar nicht zulassen. Datenbanken werden also erst im Bereich der sogenannten Personal-Computer möglich, die dann aber auch dementsprechend teuer sind.



In diesem Beispiel wird der Datensatz Nr. 1 einer Adressendatei gezeigt. Sie erkennen die Begriffe Datensatz, Datensatznummer, Datenfeld und Datenfeldnummer. Außerdem ist hier eine Eingabemaske abgebildet, nach der später die Eingaben erfolgen können.

Eine Ansammlung solcher Datensätze nennt man Datei. Unserer Datei wurde der Name "Kundenanschriften" zugeordnet.

Jeder Datensatz hat als Zugriffsschlüssel eine Nummer, unter der alle anderen Angaben aus dem Datensatz wiederzufinden sind. Wenn unter der gleichen Nummer in einer zweiten Datei noch Umsätze gespeichert würden und man auf beide Dateien gleichzeitig zugreifen könnte, könnte man im Prinzip von einer kleinen Datenbank sprechen.

Ein Datensatz ist mit einer Karteikarte vergleichbar, die auf dem Computer läuft. Ein Karteikasten wäre auf dem Computer demnach eine Datei.

Wie wir bereits aus den ersten Kapiteln dieses Buchs wissen, werden Daten, je nach Typ, in alphanumerischen und numerischen Variablen gespeichert. Dies ist bei kleinen Datenmengen kein Problem, da genügend Variablennamen zur Verfügung stehen. Stellen Sie sich aber einmal folgendes vor: Sie sollen ein Programm schreiben, das für Sie 1000 komplette Anschriften verwaltet. Eine Anschrift soll dabei aus folgenden Teilen bestehen:

a) dem Vor- und Nachnamen
b) der Straße und Hausnummer
c) der Postleitzahl
d) dem Wohnort
e) der Telefonnummer

Eine Anschrift setzt sich dann bereits aus fünf einzelnen Datenangaben zusammen. Bei 5 Datenfeldern pro Datensatz ergeben sich rechnerisch 5000 Datenfelder bei 1000 Anschriften. Wenn Sie jetzt alle verfügbaren Variablen betrachten, werden Sie feststellen, daß diese nicht mehr ausreichen. Es muß also eine andere Lösung für dieses Problem geben, und die bietet sich in der Verwendung eines neuen Variablentyps, den sogenannten "indizierten Variablen" an, mit denen wir uns im nächsten Abschnitt beschäftigen wollen.

9.1.1 Indizierte Variablen

Bevor wir uns mit dieser Thematik im einzelnen auseinandersetzen, zunächst wieder ein kleines Beispiel. Die anfangs erwähnte Adressenverwaltung soll noch einmal angesprochen werden, aber dieses Mal nur 100 Anschriften verwalten. Eine Tabelle für unsere Adressenverwaltung könnte so aussehen:

```
lfd. Nr.  Vorname  Nachname  Str. und Nr.  PLZ  Ort  Telefon
1
2
 .
 .
 .
100
```

Wie Sie sehen, hat jede Anschrift eine Nummer, der die entsprechenden Adressen-Angaben folgen. Das nächste Beispiel verdeutlicht das Grundprinzip dieser Adressenspeicherung noch einmal.

```
10    INPUT"Vorname              ";V$
20    INPUT"Nachname             ";N$
30    INPUT"Straße und Hausnr.   ";S$
40    INPUT"Postleitzahl         ";P$
50    INPUT"Wohnort              ";W$
60    INPUT"Telefon              ";T$
90    GOTO 10
```

Werden mit Hilfe dieses Programms Daten eingegeben, so werden diese beim nächsten Durchlauf des Programms wieder mit der neuen Eingabe überschrieben.

Sehen wir uns für unsere nächsten Überlegungen Zeile 10 an:

```
10 INPUT"Vorname   ";V$
```

Um das Problem des Überschreibens auszuschließen, könnte theoretisch unsere Variable so aussehen:

```
10 INPUT"Vorname   ";V$1
```

Beim nächsten Mal steht anstelle der 1 eine 2 etc. Ein solches Vorgehen ist zwar denkbar, wird in der Praxis aber anders gehandhabt. Dort arbeitet man nämlich mit Zahlen, die in Klammern hinter der Variablen stehen. Zeile 10 könnte dann so aussehen:

```
10 INPUT"Vorname   ";V$(1)
```

Kehrt das Programm durch den Befehl **GOTO** wieder in Zeile 10 zurück, würde wieder unter der Bezeichnung V$(1) ein neuer Vorname gespeichert.

Wir müssen also aus der 1 in der Klammer eine 2 machen. Am einfachsten wäre ein Zählvorgang, der jedesmal aktiviert wird, wenn eine Anschrift komplett eingegeben worden ist. Dazu müssen wir aber aus der in Klammern stehenden Angabe eine Variable machen.

Gehen wir davon aus, daß unser Zähler die Variable Z hat, können wir unsere Zeile 10 so schreiben:

```
10 INPUT"Vorname   ";V$(Z)
```

Den Zähler kann man vor dem Rücksprung mit **GOTO** einbauen. Ich habe dazu die Zeilennummer 70 eingebaut. Sie lautet:

```
70 Z = Z + 1
```

Ändern Sie nun alle Zeilennummern mit dem `INPUT`-Befehl so um, wie wir es soeben mit Zeile 10 getan haben (also mit dem Z in Klammern).

Sie können nun Ihr Programm mit **RUN** starten und versuchsweise drei Anschriften komplett eingeben. Nach Eingabe der dritten Anschrift unterbrechen Sie Ihr Programm mit der Taste <ESC>. Um sich jetzt die erste oder zweite Anschrift anzusehen, geben Sie im Direktmodus ein:

```
PRINT V$(1),N$(1),S$(1),P$(1),W$(1),T$(1)
```

Sie erhalten daraufhin die erste eingegebene Anschrift mit allen Angaben auf dem Bildschirm. Wenn Sie die zweite Anschrift sehen möchten, ersetzen Sie die 1 durch eine 2. Durch den Zähler sind also die Daten der letzten Anschriften nicht verlorengegangen. Da sich das Programm aber aufgrund des **GOTO**-Befehls in einer Endlosschleife befand, müßte es jedesmal mit <ESC> unterbrochen werden. Besser wäre demgemäß, wenn Sie abfragen, ob ein bestimmter Endwert schon erreicht wurde oder nicht.

Um beispielsweise 10 Anschriften einzugeben, fügen Sie noch die Zeile 80 hinzu.

```
80 IF Z=10 THEN END
```

Nach Eingabe der zehnten Anschrift wird das Programm automatisch beendet, ohne daß Sie die <ESC>-Taste drücken müssen. Wollen Sie die anfangs erwähnten 100 Anschriften erfassen, müssen Sie nur den Wert 10 in Zeile 80 durch den Wert 100 ersetzen.

Es gibt jedoch noch einen wesentlich einfacheren Weg: In den letzten Kapiteln lernten Sie die Programmierung von Schleifen kennen. Die Vorteile dieser Programmierungsart wollen wir uns hier zunutze machen und die Befehle FOR...NEXT einsetzen. Ändern Sie nun Ihr Beispiel so ab, daß es aussieht wie das Listing:

```
5       FOR Z = 1 TO 10
10      INPUT"Vorname       ";V$(Z)
20      INPUT"Nachname      ";N$(Z)
30      INPUT"Str. und Nr.  ";S$(Z)
40      INPUT"Postleitzahl  ";P$(Z)
50      INPUT"Wohnort       ";W$(Z)
60      INPUT"Telefon       ";T$(Z)
70      NEXT Z
```

Sie haben jetzt ein kleines Programm erstellt, das Ihnen 10 Anschriften speichert. Da Sie sicherlich auch die Ausgabe auf dem Bildschirm sehen möchten, hier der Programmteil, der alle Anschriften wieder sichtbar macht:

```
80      REM - - - -  Datenausgabe  - - - -
90      FOR I=1 TO 10
100     PRINT"Anschrift-Nr.: ";I
110     PRINT
120     PRINT V$(I);" ";N$(I)
130     PRINT S$(I)
140     PRINT P$(I);" ";W$(I)
150     PRINT T$(I)
160     E$=INKEY$:IF E$="" THEN 160
170     PRINT"============================"
180     PRINT
190     NEXT I
```

Dieses Programm druckt Ihnen die Nummer der gespeicherten Anschrift aus und setzt den Vor- und Nachnamen sowie die Postleitzahl und den Wohnort hintereinander auf den Bildschirm. In Zeile 160 wird die Tastatur abgefragt, ob eine Taste gedrückt wurde. Wenn ja, erscheint die nächste Adresse auf dem Bildschirm.

Damit kennen Sie die Arbeitsweise der indizierten Variablen und sollten diese auch anwenden können!

9.1.2 Eindimensionale Felder

Im Zusammenhang mit indizierten Variablen spricht man häufig von "Feldern". Dabei unterscheidet man ein-, zwei- oder dreidimensionale Felder. Alle Felder lassen sich direkt ansprechen. Nachfolgend ein kleines Beispiel für ein eindimensionales Feld.

```
10    CLS
20    DATA Thomas, Gerhard, Jürgen, Rolf, Ulrike
30    FOR I=1 TO 5
40    READ A$
50    NAME$(I)=A$
60    NEXT I
70    REM - - - - - - eindimensionales Feld ausgeben - - - -
80    FOR I=1 TO 5
90    PRINT"Feld-Nr.: ";I;"  Inhalt dieses Feldes: ";NAME$(I)
100   NEXT I
```

Dieses Programm liest die fünf Namen in ein Feld und gibt sie anschließend wieder aus. Die Tabelle dazu sieht so aus:

Feld-Nr.	**NAME$**
1	Thomas
2	Gerhard
3	Jürgen
4	Rolf
5	Ulrike

Auf jedes Feld können Sie direkt zugreifen. Geben Sie versuchsweise einmal im Direktmodus ein:

```
PRINT NAME$(1)
```

Nach Drücken der Taste <ENTER> erscheint der Name "Thomas" auf dem Bildschirm. Ebenso verhält es sich mit den anderen Namen, wenn Sie in Klammern die entsprechende Feldnummer einsetzen.

Natürlich lassen sich auch numerische Werte speichern, wie Sie im folgenden Programm sehen können. Es erzeugt 100 Zufallszahlen und speichert sie in indizierten Variablen ab. Lassen Sie sich überraschen, was bei dem Programm passiert!

```
10    MODE 1
20    INK 0,0
30    BORDER 0
40    CLS
50    FOR I = 1 TO 100
```

```
60    ZUFALL(I)=INT(RND(1)*1000)
70    NEXT I
80    REM - - - - - Ausgabe der Zufallszahlen - - - - -
90    FOR I=1 TO 100
100   PRINT"In Feld ";I;" steht die Zufallszahl ";ZUFALL(I)
110   NEXT I
```

Wenn Sie das Programm gestartet haben, arbeitet es zuerst einwandfrei, bricht dann aber mit der Meldung "*Subcript out of range in 60*" das Programm ab. Die Fehlermeldung gibt an, daß die angesprochene Variable ZUFALL(I) außerhalb eines vereinbarten Bereichs liegt.

Grundsätzlich wird für indizierte Variablen gleich nach dem Einschalten des Rechners vom Betriebssystem automatisch Platz bis Feld-Nr. 10 bereitgestellt. Wird mehr Platz benötigt, muß dies dem Computer mitgeteilt werden. Dafür gibt es den Befehl DIM, der eine Variable entsprechend groß dimensioniert.

Damit unser Programm auch läuft, fügen Sie Zeile 45 ein:

```
45 DIM ZUFALL(100)
```

Eindimensionale Felder benutzt man häufig, um beispielsweise Überschriften in Schleifen später aufzurufen, wie Sie in Kapitel 9.2.5 sehen werden.

9.1.3 Zweidimensionale Felder

Diese Felder dienen dazu, größere Datenmengen, die in einem bestimmten Zusammenhang zueinander stehen, zu verknüpfen und zu bearbeiten. Als Beispiel möchte ich Ihnen hier ein Modell für einen Stundenplan vorstellen. Der Stundenplan enthält für jeden Tag 8 Unterrichtsstunden und insgesamt 6 Unterrichtstage. Sehen wir uns dies einmal genauer an:

Tag	1. Ust.	2. Ust.	3. Ust.	4. Ust.	5. Ust.	6. Ust.	7. Ust.	8. Ust.
1								
2								
3								
4								
5								
6								

Jedes Feld hat eine Nummer. Der Platz für die erste Unterrichtsstunde am Montag (1) hat den Namen 1,1. Wenn wir alle Plätze als Fächer bezeichnen wollen, lautet der Platz für die 1. Stunde am Montag morgen: FACH(1,1).

Im nun folgenden Programm werden die Fächer als Zahlen dargestellt, da sich Zahlen einfacher simulieren lassen. Die Zahlen sind dann die Schlüssel zu den jeweiligen Fächern. Das Programm erstellt einen zufälligen Stundenplan. Dieses Modell dient zur Erläuterung der Arbeit mit indizierten Variablen und zweidimensionalen Feldern.

Es bedeuten:

1	=	Deutsch
2	=	Englisch
3	=	Französisch
4	=	Latein
5	=	Mathematik
6	=	Sport
7	=	Religion
8	=	Geschichte
9	=	Wirtschaftskunde
10	=	Informatik-AG
11	=	Werken
12	=	Handarbeiten
13	=	Freistunde
14	=	Biologie
0	=	kein Unterricht

Das Programm:

```
10    MODE 2
20    BORDER 0
30    INK 0,0
40    CLS
50    DATA Montag, Dienstag, Mittwoch, Donnerstag, Freitag, Samstag
60    FOR I=1 TO 6
70    READ TAGE$
80    T$(I)=TAGE$
90    NEXT I
100   DIM FACH(6,8)
110   FOR TAGE=1 TO 6: REM   für Montag bis Samstag
120   SCHLUSS=INT(RND(1)*(8-4)+4)
130   FOR STUNDE=1 TO SCHLUSS
140   FACH(TAGE,STUNDE)=INT(RND(1)*(14-1)+1)
150   NEXT STUNDE
160   NEXT TAGE
170   CLS
180   FOR TAGE=1 TO 6
190   PRINT"Tag: ";T$(TAGE)
200   PRINT
210   FOR STUNDE=1 TO 8
220   PRINT"U.-Std. ";STUNDE;"  Fach-Nr.: ";FACH(TAGE,STUNDE)
```

```
230   NEXT STUNDE: PRINT : PRINT
240   E$=INKEY$: IF E$="" THEN 240 ELSE NEXT TAGE
```

Programmbeschreibung:

Zeile 10 bis 40: Bildschirm und Farbe einstellen
Zeile 50: Wochentage zum Lesen bereitstellen
Zeile 60: Schleife zum Lesen öffnen
Zeile 70: Ersten Wochentag lesen
Zeile 80: Wochentag in eine eindimensionale Tabelle bringen
Zeile 90: Schleife wieder schließen
Zeile 100: Fächer dimensionieren (1. Angabe=Tage, 2. Angabe= U.-Std.)
Zeile 110: Schleife für 6 Wochentage wieder öffnen
Zeile 120: Zufallszahl erzeugen, wieviele Unterrichtsstunden an dem aktuellen Wochentag anfallen.
Die Zahl der Stunden liegt zwischen 4 und 8 Std.
Zeile 130: Schleife zum Erzeugen der Fächer öffnen
Zeile 140: Zufälliges Fach ermitteln und in ein zweidimensionales Feld transportieren
Zeile 150: Schleife für die Stunden wieder schließen
Zeile 160: Schleife für die Wochentage wieder schließen
Zeile 170: Bildschirm löschen
Zeile 180: Schleife für die Ausgabe von 6 Wochentagen öffnen
Zeile 190: Ausgabe, um welchen Wochentag es sich handelt
Zeile 200: Leerzeile ausgeben
Zeile 210: Schleife für die Ausgabe von maximal 8 Unterrichtsstunden öffnen
Zeile 220: Ausgabe, bei welchem Tag in der wievielten Stunde welches Fach anfällt
Zeile 230: Schleife für die Stunden wieder schließen und zwei Leerzeilen ausgeben
Zeile 240: Tastaturabfrage und Schließen der Schleife

Erweiterungen:

Denkbar wäre auch noch ein Programmteil, der die Umwandlung der Nummern vornimmt und die Unterrichtsfächer mit der richtigen Bezeichnung ausgibt.

9.2 Anwendungsbeispiele mit indizierten Variablen

In diesem Abschnitt geht es um die Anwendung indiziierter Variablen. Gezeigt werden einige Anwendungen, die von allgemeinem Interesse sind und auch in eigene Programmentwicklungen übernommen werden können. Dazu unser erstes Beispiel.

Beispiel 1: Berechnen von Quadratwurzeln

Aufgabe:

Die Quadratwurzeln der Werte 1 bis 144 sollen berechnet und in einer eindimensionalen Matrix gespeichert werden.

Eingabe:

Keine, wird vom Programm vorgegeben

Verarbeitung:

Bereich ausreichend dimensionieren
Berechnung und Zuweisung mit der Funktion SQR(I)
Verwendete Variable zur Speicherung: A(I)

Ausgabe:

Alle Ergebnisse in der folgenden Form:
Die Wurzel aus..... ist ....

Das Programm:

```
10    MODE 1
20    INK 0,0
30    BORDER 0
40    CLS
50    DIM A(144)
60    FOR I=1 TO 144
70    A(I)=SQR(I)
80    NEXT I
90    CLS
100   FOR I=1 TO 144
110   PRINT"Die Wurzel aus";I;"ist";A(I)
120   E$=INKEY: IF E$="" THEN 120
130   NEXT I
140   PRINT"Ende des Programms"
```

Programmbeschreibung:

Zeile 10 bis 40: Bildschirmmodus und Farbe einstellen
Zeile 50: Feld mit 144 Plätzen reservieren

Zeile 60:	Schleife öffnen von 1 bis 144
Zeile 70:	Das Ergebnis der Quadratwurzel in der Variablen A(I) abspeichern
Zeile 80:	Schleife wieder schließen
Zeile 90:	Bildschirm löschen
Zeile 100:	Schleife zur Datenausgabe öffnen
Zeile 110:	Aktuellen Wert und die berechnete Quadratwurzel in einen Text setzen
Zeile 120:	Tastaturabfrage, ob eine Taste gedrückt wurde
Zeile 130:	Schleife schließen
Zeile 140:	Text auf dem Bildschirm ausgeben

Änderungsmöglichkeiten:

Mit dem Befehl LOCATE können Sie die Ausgabe auf eine definierte Position bringen.

Beispiel:

```
105 LOCATE 1,10
```

Den Berechnungsbereich können Sie ändern, indem Sie die Endwerte der Schleife erhöhen. Beachten Sie dabei, daß Sie in diesem Fall höher dimensionieren müssen. Denkbar wäre, die Endwerte in einer Variablen zwischenzuspeichern, was so aussehen könnte:

```
45    INPUT"Bis zu welcher Zahl berechnen  ";Z
50    DIM A(Z)
60    FOR I=1 TO Z
100   FOR I=1 TO Z
```

Diese Möglichkeit gibt es also auch.

Beispiel 2: Summe eines Bereichs ermitteln

Aufgabe:

Ermitteln der Summe eines frei wählbaren Bereichs

Eingabe:

Anfangswert des zu verarbeitenden Bereichs
Endwert des zu verarbeitenden Bereichs

Verarbeitung:

Angegebenen Bereich abarbeiten mit der folgenden Berechnung: SUMME = SUMME + BETRAG(I)

Ausgabe:

Angabe, in welchem Bereich gearbeitet wurde
Gesamtbetrag dieses Bereichs

Das Programm:

```
10    MODE 1
20    BORDER 0
30    INK 0,0
40    CLS
50    DIM BETRAG(100)
60    FOR I=1 TO 100
70    BETRAG(I)=INT(RND(1 )*1000)
80    NEXT I
90    REM - - - Bereich mit Zufallszahlen  gefüllt - - -
100   INPUT"Anfangspunkt der Berechnung: ";A
110   INPUT"Endpunkt der Berechnung:     ";E
120   FOR I=A TO E
130   SUMME=SUMME+BETRAG(I)
140   LOCATE 1,10:PRINT"Summe jetzt: ";SUMME
150   NEXT I
160   PRINT"Der Bereich von";A;"bis";E;"ergibt  die Summe";SUMME
170   INPUT"Weitere Bereiche berechnen ?";E$
180   IF E$="j" THEN GOTO 100
190   IF E$="n" THEN END
200   PRINT"Falsche Eingabe! Nur j oder n  eingeben"
210   GOTO 170
```

Programmbeschreibung:

Zeile 10 bis 40: Farbe und Modus einstellen
Zeile 50: Platz für 100 Werte dimensionieren
Zeile 60: Schleife für 100 Berechnungen öffnen
Zeile 70: Zufallszahl zwischen 1 und 1000 erzeugen und speichern
Zeile 80: Schleife wieder schließen
Zeile 90: Kommentarzeile
Zeile 100: Frage nach dem Anfangspunkt des Bereichs
Zeile 110: Frage nach dem Endpunkt des Bereichs
Zeile 120: Schleife für diesen Bereich öffnen
Zeile 130: Berechnung durchführen nach der Formel:
Neue Summe = Alte Summe + Aktueller Betrag
Zeile 140: Textcursor positionieren und laufende Summe ausgeben
Zeile 150: Schleife für diese Berechnung wieder schließen
Zeile 160: Schlußsatz ausgeben, innerhalb welchen Bereichs gearbeitet wurde

Zeile 170: Frage, ob weitere Bereiche bearbeitet werden sollen
Zeile 180: Abfrage, ob mit ja geantwortet wurde
Zeile 190: Abfrage, ob mit nein geantwortet wurde
Zeile 200: Wenn beides nicht zutraf, handelt es sich um eine Fehleingabe, die entsprechend behandelt wird
Zeile 210: Rücksprung zur erneuten Eingabe

Bemerkungen zum Programm:

Obiges Beispiel könnte man einsetzen, wenn man den Wert einer Sammlung feststellen möchte oder in einer Lagerverwaltung die Restlängen an Rohmaterial in Metern ermitteln möchte, um festzustellen, ob für einen Auftrag noch genügend Material zur Verfügung steht oder neues Rohmaterial bestellt werden muß. In unserem Beispiel wurde dieser Abfragebereich mit Zufallszahlen gefüllt. In der Praxis arbeitet man natürlich mit echten Werten.

Beispiel 3: Suchen eines Teilnehmers über seine Telefonnummer

Aufgabe:

Aus einem Telefonverzeichnis soll ein Teilnehmer herausgesucht werden, wobei die Suche nicht über den Namen, sondern über die Telefonnummer erfolgt.

Eingabe:

Telefonnummer

Verarbeitung:

Mit einer Schleife werden Schritt für Schritt alle Inhalte der indizierten Variablen auf die gesuchte Telefonnummer hin untersucht. Wenn die gesuchte Telefonnummer nicht gefunden wird, soll das Programm die Meldung ausgeben:

```
"Gesuchte Telefonnummer nicht gefunden".
```

Enthält das Programm die Telefonnummer, sollen Nummer und Name des Teilnehmers auf dem Bildschirm ausgegeben werden.
Danach soll das Programm fragen, ob ein weiterer Name gesucht werden soll. Wenn ja,beginnt das Programm erneut, wenn nein, wird es beendet.

Ausgabe:

Gefundene Telefonnummer und der dazugehörige Name

Das Programm:

```
10    MODE 2
20    INK 0,0
30    BORDER 0
40    CLS
50    I$=CHR$(24)
60    DIM TELNR$(12), NAME$(12)
70    DATA 05241-15182, Thomas Erpel
80    DATA 0521-173133, Ulrike Münstermann
90    DATA 0211-131345, Walter Heimer
100   DATA 0435-1267, Werner Germersheim
110   DATA 08745-3457, Paul Stopka
120   DATA 02356-3434, Alfred Noack
130   DATA 05521-12399, Autohaus Mense KG
140   DATA 0753-4545, Meisterbau GmbH
150   DATA 0645-3040, Software AG
160   DATA 05214-5656, Baumaschinen GmbH
170   DATA 0789-231177, Glasbau Friedrich GmbH
180   DATA 089-45569, Malerbedarf Allersmann
190   FOR I=1 TO 12
200   READ TELEFONN$, NAME$
210   TELNR$(I)=TELEFONN$
220   NAME$(I)=NAME$
230   NEXT I
240   CLS
250   FOR I=1 TO 12
260   PRINT"Rufnummer: ";TELNR$(I);TAB(30); "Name: ";NAME$(I)
270   PRINT
280   NEXT I
290   PRINT STRING$(79,154)
300   PRINT"Diese Namen befinden sich jetzt in Ihrem Verzeichnis"
310   FOR T=1 TO 2000:NEXT T:CLS
320   INPUT"Welche Telefonnummer wollen Sie eingeben ";T$
330   FOR I=1 TO 12
340   IF TELNR$(I)=T$ THEN GOTO 390
350   NEXT I
360   PRINT
370   PRINT I$;" Gesuchte Telefonnummer befindet sich nicht im Verzeichnis";I$
380   GOTO 420
390   CLS
400   PRINT"Der gesuchte Teilnehmer heisst: "; I$;" ";NAME$(I);" ";I$;"und
      hat"
410   PRINT"die Rufnummer: ";I$;" ";TELNR$(I);"  ";I$
420   PRINT
430   PRINT"Nochmals nach einem Teilnehmer suchen ??"
440   E$=INKEY$: IF E$="" THEN 440
450   IF E$="j" THEN CLS:GOTO 320
460   IF E$="n"THEN END
470   GOTO 440
```

Programmbeschreibung:

Zeile 10 bis 40: Bildschirm und Farben setzen

Zeile 50: Der Variablen I$ den ASCII-Code für REVERS zuordnen

Zeile 60: Die Variablen für die Telefonnummer und den Namen ausreichend dimensionieren

Zeile 70 bis 180: DATA-Zeilen mit verschiedenen Telefonnummern und den jeweiligen Teilnehmern

Zeile 190: Schleife zum Lesen der Daten öffnen

Zeile 200: Die Telefonnummer und den Namen einlesen

Zeile 210: Die eben eingelesene Telefonnummer in ein Datenfeld transportieren

Zeile 220: Den eben gelesenen Namen in ein Datenfeld transportieren

Zeile 230: Diese Schleife wieder schließen

Zeile 240: Bildschirm löschen

Zeile 250: Schleife zur Datenausgabe öffnen

Zeile 260: Rufnummer und Teilnehmer auf dem Bildschirm ausgeben

Zeile 270: Leerzeile auf dem Bildschirm ausgeben

Zeile 280: Schleife für die Ausgabe wieder schließen

Zeile 290: Linie, bestehend aus 79 Zeichen des ASCII-Codes 154, auf dem Bildschirm ausgeben

Zeile 300: Text unter diese Übersicht setzen

Zeile 310: Warteschleife aktivieren und nach Ablauf der Schleife den Bildschirm löschen

Zeile 320: Eingabe der Telefonnummer, die als Suchbegriff dienen soll

Zeile 330: Schleife für die Suche öffnen

Zeile 340: Abfrage, ob die aktuelle Telefonnummer gleich der gesuchten Telefonnummer ist.
Wenn ja, dann Zeile 390 anspringen

Zeile 350: Schleife für die Suche wieder schließen

Zeile 360: Leerzeile auf den Bildschirm drucken

Zeile 370: Ausgabe der Nachricht "Gesuchte Telefonnummer befindet sich nicht im Verzeichnis" in reverser Darstellung

Zeile 380: Sprung in Zeile 420 (Abfrage, ob nochmals gesucht werden soll)

Zeile 390: Bildschirm löschen

Zeile 400: Ausgabe des Teilnehmers und der Telefonnummer in reverser Darstellung

Zeile 410: wie Zeile 400

Zeile 420: Leerzeile auf Bildschirm drucken

Zeile 430: Text ausgeben "Nochmals nach einem Teilnehmer suchen?"

Zeile 440: Abfrage, ob irgendeine Taste auf der Tastatur gedrückt wurde

Zeile 450: Wenn die Taste j gedrückt wurde, dann Bildschirm löschen und zur Zeile 320 gehen

Zeile 460: Wenn die Taste n gedrückt wurde, dann das Programm mit dem Befehl END beenden

Zeile 470: Rücksprung in Zeile 440 (erneute Abfrage der Tasten)

Bemerkungen zum Programm:

Für Zeile 50 wurde mitgeteilt, daß der ASCII-Code 24 (REVERS) der Variablen I$ zugewiesen wird. Die Verwendung von I$ wirkt wie ein Schalter. Wird I$ das erste Mal aufgerufen, schaltet der Rechner automatisch in die reverse Darstellung um, bis zum zweiten Mal I$ gefunden wird. Dies ist einfacher, als jedesmal CHR$(24) zu schreiben.

Sie können das ganze Programm auch so umbauen, daß Sie einen Namen eingeben und der Computer Ihnen dazu den passenden Anschluß heraussucht. Eine Erweiterung ist dadurch möglich, daß Sie zusätzliche DATA-Zeilen eingeben und die Endwerte der Schleife höher setzen. Vergessen Sie dabei nicht die Anpassung der DIM-Anweisungen in Zeile 60!

Beispiel 4: Durchschnittstemperatur ermitteln

Aufgabe:

Ermitteln der Durchschnittstemperatur des Monats Januar anhand von vorliegenden Meßwerten. Die Meßwerte in diesem Beispiel werden durch den Zufallsgenerator simuliert. Die so erzielten Temperaturen liegen zwischen minus 20 und plus 11 Grad Celsius. Insgesamt wurde im Monat Januar stündlich eine Messung vorgenommen, um ein möglichst genaues Ergebnis zu erhalten. Bei 31 Tagen ergibt dies rechnerisch 31*24 Messungen.

Eingabe:

Keine, weil sie vom Programm fest vorgegeben wird

Verarbeitung:

Hier muß wieder schrittweise vorgegangen werden:

1. Schritt: Dimensionierung der Variablen
2. Schritt: Simulation der Temperaturen zwischen -20 und +11 Grad Celsius

3. Schritt: Simulierte Daten in indizierten Variablen abspeichern
4. Schritt: Summe aller Temperaturen bilden und den Mittelwert errechnen
5. Schritt: Ergebnis präsentieren

Ausgabe:

Alle Temperaturen in Tabellenform sowie das ermittelte Ergebnis

Das Programm:

```
10    MODE 2
20    INK 0,0
30    BORDER 0
40    CLS
50    DIM TEMPERATUR(750)
60    REM - - - - Insgesamt 744 Messwerte simulieren - - - -
70    LOCATE 10,10:PRINT"Einen Augenblick. Berechnung läuft...."
80    FOR I=1 TO 744
90    RANDOMIZE TIME
100   TEMPERATUR(I)=(RND(1)*(-21-10)+11
110   TEMPERATUR(I)=INT(TEMPERATUR(I)+10+.5)/10
120   LOCATE 10,15
130   PRINT"Messwert-Nr.:";I;"wird simuliert."
140   NEXT I
150   REM
160   REM - - - - - Durchschnitt berechnen - -
170   REM
180   FOR I=1 TO 744
190   SUMME=SUMME+TEMPERATUR(I)
200   NEXT I
210   MITTELWERT=SUMME/744
220   MITTELWERT=INT(MITTELWERT*10+.5)/10
230   REM
240   REM - - - - - Ausgabe der Daten - - - - -
250   CLS
260   FOR I=1 TO 744
270   PRINT"Messwert-Nr.:";I;"ergab "; TEMPERATUR(I);TAB(32);"Grad Celsius"
280   FOR T=1 TO 100:NEXT T
290   NEXT I
300   PRINT
310   PRINT CHR$(24);"Die Durchschnittstemperatur betrug im Monat
      Januar ";
320   PRINT MITTELWERT;" Grad Celsius.";CHR$(24)
```

Programmbeschreibung:

Zeile 10 bis 40: Modus und Farbe setzen
Zeile 50: Die Variable TEMPERATUR mit 750 Plätzen dimensionieren
Zeile 60: Kommentarzeile
Zeile 70: Textcursor positionieren und Text ausgeben
Zeile 80: Schleife für die Simulation öffnen
Zeile 90: Zufallsgenerator neu generieren

Zeile 100:	TEMPERATUR(I) mit einem zufälligen Wert zwischen -20 und +11 Grad Celsius füllen
Zeile 110:	Temperatur auf eine Nachkommastelle runden
Zeile 120:	Textcursor neu positionieren
Zeile 130:	Angabe, welcher Meßwert gerade simuliert wird
Zeile 140:	Schleife für die Simulation wieder schließen
Zeile 150:	Kommentarzeile
Zeile 160:	Kommentarzeile
Zeile 170:	Kommentarzeile
Zeile 180:	Schleife für die Addition der Temperaturen öffnen
Zeile 190:	Neue Summe=Alte Summe+aktuelle Temperatur
Zeile 200:	Schleife wieder schließen
Zeile 210:	Mittelwert berechnen
Zeile 220:	Mittelwert auf eine Nachkommastelle runden
Zeile 230:	Kommentarzeile
Zeile 240:	Kommentarzeile
Zeile 250:	Bildschirm löschen
Zeile 260:	Schleife für die Ausgabe aller Daten öffnen
Zeile 270:	Aktuellen Meßwert und dessen Nummer ausgeben
Zeile 280:	`FOR T=1 TO 100:NEXT` ergibt eine Warteschleife
Zeile 290:	Schleife für die Datenausgabe wieder schließen
Zeile 300:	Leerzeile auf dem Bildschirm ausgeben
Zeile 310:	Ausgabe der Durchschnittstemperatur - Zeile 1
Zeile 320:	Ausgabe der Durchschnittstemperatur - Zeile 2 Zeilen 310 und 320 werden revers ausgegeben

Bemerkungen zum Programm:

Bei dem Beispiel der Temperaturermittlung handelt es sich um eine reine Simulation; das Programm läßt sich auf viele andere Bereiche übertragen.

Beispiel 5: Umsatzberechnung

Aufgabe:

Die Problemstellung lautet: Ein Unternehmen der Textilbranche besitzt in den Städten Berlin, Bonn, Bielefeld, Dortmund und Hannover je ein Geschäft. Um alle Geschäfte umsatzmäßig vergleichen zu können, benötigt man die Umsätze für jedes Geschäft und jeden Kalendertag. Früher wurde das gewünschte Zahlenmaterial mühsam von Hand erstellt. Mit Hilfe eines Computers soll die

Ausrechnung von Hand entfallen. Aufgabe ist, für dieses Unternehmen ein brauchbares Programm zu entwickeln. Folgende Forderungen werden dabei angemeldet.

Es soll möglich sein:

a. den Gesamtumsatz aller Geschäfte pro Monat automatisch zu ermitteln
b. den Tagesumsatz aller Geschäfte zu errechnen
c. den monatlichen Umsatz einzelner Geschäfte zu berechnen
d. einen Umsatzvergleich anzustellen, bei dem das Programm nach Umsatzhöhe sortiert wird

Eingabe:

Originalumsätze für jedes Geschäft pro Kalendertag

Verarbeitung:

siehe nächste Seite (Tabellenübersicht)

Ausgabe:

Geforderte Daten in übersichtlicher Form

Hinweis:

Da das Programm auch vom Verkaufspersonal bedient werden soll, entscheidet man sich für ein Menü als Benutzerführung.

```
Tabellenübersicht "UMSATZBERECHNUNG"
Kalendertag  G E S C H Ä F T E
             Berlin  Bielefeld Bonn  Dortmund Hannover
Umsatz pro
      (1)   (2)     (3)     (4)     (5)      Tag ges.
1     1,1   1,2     1,3     1,4     1,5
2     2,1   2,2     2,3     2,4     2,5
3     3,1   3,2     3,3     3,4     3,5
4     4,1   4,2     4,3     4,4     4,5
5     5,1   5,2     5,3     5,4     5,5
6     6,1   6,2     6,3     6,4     6,5
7     7,1   7,2     7,3     7,4     7,5
8     8,1   8,2     8,3     8,4     8,5
 ....
 ....
31    31,1  31,2    31,3    31,4    31,5
      Summe Summe   Summe   Summe   Summe    Gesamt-
                                             summe
```

Zur Information:

Jedes Feld innerhalb dieser Umsatztabelle hat eine ganz bestimmte Nummer. Diese Nummer ergibt sich aus den Zeilennummern und der

Spaltennummer. Die Felder können durch Aufrufen ihrer Nummer angesprochen werden. Im Programm haben die Nummern später eine bestimmte Bezeichnung; die Variable GESCHÄFT steht dabei für die fünf einzelnen Geschäfte und die Variable TAGE für die 31 Kalendertage.

Ein Wort noch zur Variablen TAGE. Es ist nicht erlaubt, TAG als Variablenname einzusetzen, da **TAG** ein BASIC-Befehl ist, der für die Grafik benutzt wird. Aus diesem Grund wurde die Bezeichnung TAGE gewählt. Alle Felder werden mit Hilfe von Schleifen bearbeitet, die verschachtelt sind.

Das Programm:

```
100   MODE 2
110   INK 0,0
120   BORDER 0
130   CLS
140   I$=CHR$(24)
150   REM - - - - - - - - Daten einlesen - - - - - -
160   DATA Berlin, Bielefeld, Bonn, Dortmund, Hannover
170   FOR I=1 TO 5
180   READ FILIALE$
190   FILIALE$(I)=FILIALE$
200   NEXT I
210   REM
220   DIM UMSATZ (31,5)
230   DIM SUMG(5):
240   PRINT TAB(20);"U M S A T Z - B E R E C H N U N G "
250   PRINT
260   PRINT"1 = Umsätze simulieren für 5 Geschäfte innerhalb ";
270   PRINT"eines Monats"
280   PRINT
290   PRINT"2 = Originale Daten zur Verarbeitung eingeben"
300   PRINT STRING$(79,154)
310   E$=INKEY$:IF E$="" THEN 310
320   IF E$="1" THEN GOSUB 1490: GOTO 520:
330   IF E$="2" THEN GOTO 370
340   SOUND 1,600,50,7
350   SOUND 2,601,50,7
360   GOTO 310
370   REM  - - - - Eingabeteil- - - - -
380   FOR TAGE=1 TO 31
390   CLS
400   PRINT I$
410   PRINT"Tagesumsätze für den";TAGE;".ten Tag eingeben:"
420   PRINT I$:
430   FOR GESCHAEFT= 1 TO 5
440   PRINT: PRINT
450   PRINT"Bitte geben Sie den Tagesumsatz für die Filiale in ";
460   PRINT FILIALE$(GESCHAEFT):" ein: ";
470   INPUT UMSATZ(TAGE,GESCHAEFT)
480   PRINT STRING$(79,154)
490   NEXT
```

```
500   NEXT:
510   REM - - - - - - - - - - - -
520   CLS
530   PRINT" Auswertungsmodus "
540   PRINT STRING$(79,154)
550   PRINT
560   PRINT"1 = Gesamtumsatz aller Geschäfte pro Monat  berechnen"
570   PRINT
580   PRINT"2 = Tagesumsätze aller Geschäfte berechnen"
590   PRINT
600   PRINT"3 = Monatlichen Umsatz einzelner Geschäfte berechnen"
610   PRINT
620   PRINT"4 = Umsatzvergleich aller Geschäfte "
630   PRINT
640   PRINT"5 = Programm beenden"
650   PRINT
660   PRINT
670   PRINT"Bitte wählen Sie eine Zahl zwischen 1 und 5:  ";
680   E$=INKEY$:IF E$="" THEN 680
690   ON VAL(E$) GOSUB 710,840,970,1110,1480
700   GOTO 520
710   CLS: REM - - - - - - - - - - - - - -
720   PRINT"Modus:  Gesamtumsatz berechnen"
730   PRINT STRING$(79,154)
740   PRINT
745   UMSATZ=0
760   FOR GESCHAEFT= 1 TO 5
770   UMSATZ=UMSATZ+UMSATZ(TAGE,GESCHAEFT)
780   NEXT GESCHAEFT
790   NEXT TAGE
800   PRINT"Der Umsatz aller Geschäfte beträgt in diesem Monat";
810   PRINT UMSATZ;" DM."
820   GOSUB 1570
830   RETURN
840   CLS: REM - - - - -
845   SUMME=0
850   PRINT"Modus: Tagesumsatz aller Geschäfte berechnen"
860   PRINT STRING$(79,154)
870   FOR TAGE =1 TO 31
880   SUMME=0
890   FOR GESCHAEFT=1 TO 5
900   SUMME=SUMME+UMSATZ(TAGE,GESCHAEFT)
910   NEXT
920   PRINT"Tagesumsatz für den";TAB(23);TAGE;
930   PRINT".ten Tag: ";SUMME;" DM."
940   NEXT
950   GOSUB 1570
960   RETURN
970   CLS: REM - - -
975   SUMME=0
980   PRINT"Modus: Monatsumsatz einzelner Geschäfte berechnen"
990   PRINT STRING$(79,154)
1000  FOR GESCHAEFT=1 TO 5
1010  SUMME=0
1020  FOR TAGE=1 TO 31
1030  SUMME=SUMME+UMSATZ(TAGE,GESCHAEFT)
1040  NEXT TAGE
```

```
1050  PRINT"Monatsumsatz für das Geschäft in ";FILIALE$ (GESCHAEFT);
1060  PRINT TAB(47);":";SUMME;" DM."
1070  PRINT
1080  NEXT GESCHAEFT
1090  GOSUB 1570
1100  RETURN
1110  CLS: REM - -  - - - -
1120  PRINT"Modus: Umsatz-Vergleich aller 5 Geschäfte"
1130  PRINT STRING$(79,154)
1135  FOR GET=1 TO 5:sum(G)=0:NEXT
1140  FOR G=1 TO 5
1150  FOR TAGE=1 TO 31
1160  SUM(G)=SUM(G)+UMSATZ(TAGE,G)
1170  NEXT TAGE
1180  NEXT G
1190  FOR I=1 TO 5
1200  PRINT:PRINT
1210  PRINT"Monatsumsatz in ";FILIALE$(I);TAB(26);":";
1220  PRINT SUM(I);TAB(37);"DM"
1230  NEXT I
1240  PRINT:PRINT:
1250  ZEIGER=0
1260  FOR I=1 TO 4
1270  IF SUM(I)<=SUM(I+1) THEN 1320
1280  H=SUM(I): H$=FILIALE$(I)
1290  SUM(I)=SUM(I+1) :FILIALE$(I)=FILIALE$(I+1)
1300  SUM(I+1)=H: FILIALE$(I+1)=H$
1310  ZEIGER=1
1320  NEXT
1330  IF ZEIGER=1 THEN 1250
1340  PRINT:PRINT"Wenn Sie jetzt eine Taste drücken, erscheinen "
1350  PRINT"die Umsätze sortiert......"
1360  GOSUB 1570
1370  CLS: REM - - - - - - - - -
1380  PRINT"Modus: Umsatz sortiert nach Höhe"
1390  PRINT STRING$(79,154)
1400  PRINT
1410  FOR I=5 TO 1 STEP -1
1420  PRINT"Monatsumsatz in ";FILIALE$(I);TAB(26);":";
1430  PRINT SUM(I);TAB(37);"DM"
1440  PRINT
1450  NEXT I
1460  GOSUB 1570
1470  RETURN
1480  CLS:END:REM - - - -Ende- - - -
1490  LOCATE 10,10
1495  FOR I=1 TO31:FOR G=1 TO 5:UMSATZ(I,G)=0:NEXT G:NEXT I
1500  PRINT"Simulationsvorgang läuft..........."
1510  FOR I=1 TO 31
1520  FOR G=1 TO 5
1530  UMSATZ(I,G)=(RND(1)*1000)
1540  UMSATZ(I,G)=INT(UMSATZ(I,G)*100+0.5)/100
1550  NEXT: NEXT: RETURN
1560  REM - - - - Unterprogramm - - - - -
1570  E$=INKEY$:IF E$="" THEN 1570 ELSE RETURN
```

Programmbeschreibung:

Zeile 100 bis 130: Bildschirm und Farben einstellen
Zeile 140: ASCII-Code für Reversdarstellung zuweisen
Zeile 150: Kommentarzeile
Zeile 160: Daten der verschiedenen Geschäfte zum Lesen bereitstellen
Zeile 170: Schleife zum Lesen öffnen
Zeile 180: Ersten Begriff lesen
Zeile 190: Begriff in Datenfeld bringen
Zeile 200: Schleife wieder schließen
Zeile 210: Kommentarzeile
Zeile 220: Umsatzfelder dimensionieren
Zeile 230: Felder für Summe/Geschäft dimensionieren
Zeile 240: Textausgabe
Zeile 250: Leerzeile auf dem Bildschirm erzeugen
Zeile 260: Textausgabe
Zeile 270: Textausgabe
Zeile 280: Leerzeile auf dem Bildschirm erzeugen
Zeile 290: Textausgabe
Zeile 300: Linie aus 79 Grafikzeichen vom ASCII-Code 154 zeichnen
Zeile 310: Tastaturabfrage
Zeile 320: Abfrage, ob der Simulatorvorgang gewünscht wird
Zeile 330: Abfrage, ob Originalwerte eingegeben werden sollen
Zeile 340: Bei falscher Eingabe einen tiefen Ton erzeugen
Zeile 350: Wie Zeile 340, nur Ton auf Kanal 2
Zeile 360: Rücksprung zur erneuten Abfrage der Tastatur in Zeile 310
Zeile 370: Kommentarzeile
Zeile 380: Schleife öffnen für die Eingabe von originalen Werten für 31 Kalendertage
Zeile 390: Bildschirm löschen
Zeile 400: Reversdarstellung einschalten durch Aufruf von I$
Zeile 410: Textausgabe, für den wievielten Tag die Umsätze eingegeben werden sollen
Zeile 420: Reversdarstellung wieder abschalten durch erneuten Aufruf von I$
Zeile 430: Die Schleife zur Verarbeitung der fünf Geschäfte wird eröffnet
Zeile 440: Zwei Leerzeilen auf dem Bildschirm erzeugen
Zeile 450: Textausgabe

Zeile 460: Angabe, für welches Geschäft der Umsatz eingegeben werden soll
Zeile 470: Dateneingabe über INPUT in ein zweidimensionales Datenfeld
Zeile 480: Linie aus 79 Grafikzeichen zeichnen
Zeile 490: Schleife für die Eingabe der 5 Geschäftsumsätze je Kalendertag
Zeile 500: Schleife für die Kalendertage wieder schließen
Zeile 510: Kommentarzeile
Zeile 520: Bildschirm löschen
Zeile 530: Textausgabe
Zeile 540: Linie aus 79 Grafikzeichen zeichnen
Zeile 550: Leerzeile auf dem Bildschirm erzeugen
Zeile 560 bis 670: Textausgabe und Leerzeilen
Zeile 680: Tastaturabfrage
Zeile 690: Die Variable E$ wird in einen numerischen Wert umgewandelt, mit dem dann die Sprunganweisungen in die Unterprogramme aufgerufen werden
Zeile 700: Zum Anfang des Menüs gehen
Zeile 710: Bildschirm löschen. Hier beginnt der Programmteil "Gesamtumsatz berechnen"
Zeile 720: Textausgabe
Zeile 730: Linie aus 79 Grafikzeichen zeichnen
Zeile 745: UMSATZ auf 0 setzen.
Zeile 740: Leerzeile auf dem Bildschirm ausgeben
Zeile 750: Schleife zur Verarbeitung von 31 Kalendertagen öffnen
Zeile 760: Schleife zur Verarbeitung der 5 Geschäfte öffnen
Zeile 770: Aktuellen Umsatz zur Summe-Umsatz addieren
Zeile 780: Schleife für die Geschäfte wieder schließen
Zeile 790: Schleife für die Kalendertage wieder schließen Bei den Zeilennummern 750 bis 790 handelt es sich um sogenannte verschachtelte Schleifen.
Zeile 800: Textausgabe
Zeile 810: Textausgabe (Rest von Zeile 800)
Zeile 820: Unterprogramm aufrufen (Tastaturabfrage)
Zeile 830: Mit dem Befehl RETURN wieder zurückspringen ins Auswertungsmenü. Dabei geht das Programm erst zur Zeilennummer 700 zurück und wird anschließend auf die Zeilennummer 520 geschickt.
Zeile 840: Bildschirm löschen

Hier beginnt der Programmteil "Tagesumsatz aller Geschäfte berechnen"

Zeile 845: Summe auf 0 setzen
Zeile 850: Textausgabe
Zeile 860: Linie aus 79 Grafikzeichen zeichnen
Zeile 870: Schleife zur Verarbeitung von 31 Kalendertagen öffnen
Zeile 880: Variable SUMME auf 0 setzen, da sie für 31 Tagen benutzt wird. Fehlt diese Zeile im Programm, würde bei jedem Durchlauf der Schleife die Gesamtsumme des vorherigen Tages aufaddiert.
Zeile 890: Schleife für die 5 einzelnen Geschäfte öffnen
Zeile 900: Umsatz aller Geschäfte an diesem Kalendertag aufaddieren
Zeile 910: Schleife für die 5 Geschäfte wieder schließen
Zeile 920 und 930: Ausgabe, wie hoch der Tagesumsatz für den aktuellen Tag ist
Zeile 940: Schleife für Kalendertage wieder schließen
Zeile 950: Unterprogramm "Tastaturabfrage" aufrufen
Zeile 960: Rücksprung ins Auswertungsmenü
Zeile 970: Bildschirm löschen

An dieser Stelle beginnt der Programmteil "Monatsumsatz einzelner Geschäfte berechnen"

Zeile 975: Summe auf Null stellen
Zeile 980: Textausgabe
Zeile 990: Linie aus 79 Grafikzeichen zeichnen
Zeile 1000: Schleife für die Verarbeitung von 5 Geschäften öffnen
Zeile 1010: SUMME auf 0 setzen, da diese Variable auch hier mehrfach benutzt wird.
Zeile 1020: Schleife für 31 Kalendertage öffnen
Zeile 1030: Umsatz für das aktuelle Geschäft über 31 Tage aufaddieren und in der Variablen SUMME speichern
Zeile 1040: Schleife für die Kalendertage wieder schließen
Zeile 1050: Textausgabe, in welchem Geschäft dieser Umsatz erzielt wurde
Zeile 1060: Ausgabe des errechneten Umsatzes für das aktuelle Geschäft
Zeile 1070: Leerzeile ausgeben auf dem Bildschirm
Zeile 1080: Schleife für die Geschäfte wieder schließen
Zeile 1090: Unterprogramm "Tastaturabfrage" aufrufen
Zeile 1100: Rücksprung zum Auswertungsmenü
Zeile 1110: Bildschirm löschen

An dieser Stelle beginnt der Programmteil "Umsatzvergleich aller 5 Geschäfte"

Zeile 1120 bis 1230:: Dieser erste Teil des Vergleichs ist identisch mit den Zeilennummern 990 bis 1080
Zeile 1240: Zwei Leerzeilen auf dem Bildschirm ausgeben

Hier beginnt die Sortierung der Umsätze nach ihrer Höhe
Zeile 1250: ZEIGER auf 0 setzen. Dieser Zeiger zeigt an, ob eine Vertauschung vorgenommen wurde. Bevor die Schleife zur Sortierung erneut durchlaufen wird, muß der Zeiger wieder auf 0 gesetzt werden.
Zeile 1260: Schleife für die Sortierung der Umsätze öffnen
Zeile 1270: Wenn der Inhalt des aktuellen Datenfeldes kleiner oder gleich dem nächsten Datenfeld ist, dann nichts vertauschen und in 1320 die Schleife schließen.
Zeile 1280: Den aktuellen Umsatz und das aktuelle Geschäft in den Hilfsvariablen H und H$ zwischenspeichern
Zeile 1290: Den Inhalt des nächstens Feldes in das aktuelle Feld transportieren. Dies gilt für den Umsatz und für das Geschäft.
Zeile 1300: Den Inhalt der Hilfsvariablen ins nächste Feld bringen. Auch das betrifft wieder den Umsatz und das entsprechende Geschäft.
Zeile 1310: Als Kennzeichen dafür, daß sortiert wurde, wird der Zeiger auf 1 gesetzt.
Zeile 1320: Schleife wieder schließen
Zeile 1330: Abfrage, ob sortiert wurde. Wenn ja, dann den gesamten Vorgang wiederholen, wenn nein, geht es in der nächsten Zeile weiter.
Zeile 1340: Leerzeile und Textausgabe
Zeile 1350: Textausgabe
Zeile 1360: Unterprogramm "Tastaturabfrage" aufrufen
Zeile 1370: Bildschirm löschen

Hier werden jetzt die Umsätze sortiert nach ihrer Höhe ausgegeben
Zeile 1380: Textausgabe
Zeile 1390: Linie aus 79 Grafikzeichen zeichnen
Zeile 1400: Leerzeile erzeugen
Zeile 1410: Schleife zur Ausgabe der Umsätze öffnen

Hier wird rückwärts gezählt, weil der größte Umsatz sich in Feld 5 befindet und der kleinste Umsatz in Feld 1 liegt.

Zeile 1420: Textausgabe, in welchem Geschäft der folgende Umsatz erzielt wurde
Zeile 1430: Ausgabe, welcher Umsatz in dem Geschäft erzielt wurde
Zeile 1440: Leerzeile erzeugen
Zeile 1450: Schleife für die Ausgabe wieder schließen
Zeile 1460: Unterprogramm "Tastaturabfrage" aufrufen
Zeile 1470: Rücksprung ins Auswertungsmenü
Zeile 1480: Bildschirm löschen
Programm wird an dieser Stelle beendet, wenn im Menü der Punkt 5 angewählt wird.
Zeile 1490: Textcursor positionieren

Simulationsvorgang beginnt hier
Zeile 1495: Alle Variablen dazu auf Null setzen
Zeile 1500: Textausgabe
Zeile 1510: Schleife für 31 Kalendertage öffnen
Zeile 1520: Schleife für 5 Geschäfte öffnen
Zeile 1530: Umsatz über Zufallsgenerator erzeugen
Zeile 1540: Den erzeugten Umsatz noch auf zwei Nachkommastellen runden
Zeile 1550: Schleife für Geschäfte schließen
Schleife für Tage schließen
Zum Auswertungsmenü zurückgehen
Zeile 1560: Kommentarzeile
Zeile 1570: Tastaturabfrage
Wird keine Taste gedrückt, geht das Programm wieder zum Anfang der Zeile, sonst springt es mit RETURN zurück.

Bemerkungen zum Programm:

Dieses Programm ist ein ausführliches, praktisches Beispiel, das ich einmal für einen Kunden entwickelt habe. Falls ein solches Programm für Sie von Interesse ist, sollten Sie mindestens ein Diskettenlaufwerk besitzen, damit Sie mit den eingegebenen Daten später auch weiterarbeiten können. Das Originalprogramm enthielt noch einen Betriebstage-Kalender, in dem die Sonn- und Feiertage abgespeichert waren. Außerdem fehlt im vorliegenden Programm die gesamte Druckerausgabe und Datenspeicherung. Wenn Sie dieses Buch durchgearbeitet haben, wird es sicherlich für Sie kein großes Problem sein, bei Bedarf diese Programmteile selbst zu erstellen.

Den Aufbau des Programms, das in dieser Form zwar übersichtlich, aber dafür sehr lang ist, können Sie ändern, indem Sie beispielsweise

für immer wiederkehrende Vorgänge weitere Unterprogramme einbauen.

10 Definieren von Benutzerfunktionen

Für wiederholt vorkommende Berechnungen können Sie sich selbst sogenannte Benutzerfunktionen erstellen und diese dann beliebig oft aufrufen. Diese selbst definierten Funktionen arbeiten z.B. wie die mathematischen Funktionen. Wir werden uns im vorliegenden Kapitel mit der Einstellung und Anwendung solcher Funktionen beschäftigen.

Sie haben dabei die Möglichkeit, mindestens 26 Funktionen selbst zu erstellen, was für die Praxis schon mehr als ausreichend ist. Es ist also neben den Standardfunktionen noch eine Menge Spielraum für eigene Entwicklungen vorhanden. Ich selbst habe meine Experimente beim Definieren der 26. Funktion eingestellt. Und nun zu unserem ersten Beispiel:

10.1 Grundlagen beim Definieren einer Benutzerfunktion

Gehen wir einmal von folgender Ausgangssituation aus: In einem Programm wird an 25 verschiedenen Stellen eine Zeile benötigt, in der eine Zufallszahl erzeugt werden soll, die zwischen 1 und 1000 liegt und zusätzlich eine Nachkommastelle hat. Im Normalfall müßten Sie dann jedesmal die folgende Zeile schreiben:

```
Zeilennummer  ROUND(RND(1)*1000,1)+1
```

In einem Programm bedeutet es immer eine Verschwendung von Speicherplatz, wenn es alternativ auch weniger speicherintensive Möglichkeiten gibt, die obendrein auch noch anwendungsfreundlicher sind. Anwendungsfreundlich deshalb, weil die reine Berechnung, also die Formel, nur ein einziges Mal geschrieben werden muß. Bei weiteren Berechnungen im Programm braucht nur noch die Formel "aufgerufen" zu werden, was einem schon viel Schreiberei spart.

Zur Definition dieser Formel stellt der Rechner einen speziellen Befehl zur Verfügung. Er lautet DEF FN und wird folgendermaßen angewandt:

Zeilennummer DEF FN *Name der Funktion (kein, ein oder mehrere Argumente) = Eigene Formel*

Wenn wir einmal davon ausgehen, daß in Zeilennummer 100 später diese Benutzerfunktion stehen soll, sieht unsere Funktion so aus:

```
100 DEF FN ZUFALL=ROUND(RND(1)*1000,1)+1
```

Mit FN, Name der Funktion, kann man diese Formel wieder aufrufen. Wenn wir davon ausgehen, daß eine Zufallszahl in einer neuen Variablen gespeichert werden soll, könnte unsere nächste Zeile so aussehen:

```
110 Z=FN ZUFALL
```

In der Variablen Z steht nun die gewünschte Zufallszahl mit einer Nachkommastelle zur Verfügung. Und hier nochmals die Zusammenfassung:

Definieren einer Funktion mit:

```
DEF FN Funktionsname (Argument) = Formel
```

Aufruf einer Benutzerfunktion:

```
X = FN Funktionsname (Argument)
```

Die gewünschte Zahl steht jetzt in einer Variablen zur Verfügung. Nachfolgend werden einige spezielle Beispiele erörtert und detailliert erklärt. Bei vielen Programmen wird diese Möglichkeit der Funktionsdefinition leider vernachlässigt, weil der Vorgang vielleicht auf den ersten Blick etwas kompliziert aussieht.

10.2 Anwendungsbeispiele für Benutzerfunktionen

In diesem Abschnitt geht es um die Anwendung von Benutzerfunktionen und zwar vorrangig aus dem mathematischen Bereich.

Beispiel 1: Definieren der Funktion COTANGENS

Diese Funktion ist im normalen BASIC-Sprachschatz nicht vorhanden und wird nun mit einer Benutzerfunktion definiert.

```
DEF FN COT(X)=1/TAN(X)
```

Mit dem Aufruf FN COT(X) erhält man den COTANGENS eines Winkels.

Beispiel 2: Definieren der Funktion ARCUSSINUS

Auch diese Funktion fehlt im Standard-BASIC und wird wie folgt definiert:

```
DEF FN ARCS(X)=ATN(X/SQR(1-X 2))
```

Später wird die Funktion wieder aufgerufen mit der Anweisung:

```
FN ARCS(X)
```

Die Variable X enthält dabei den gewünschten Wert, den man umwandeln möchte.

Beispiel 3: Definieren der Funktion ARCUSCOTANGENS

Auch diese Funktion läßt sich wieder mit einem DEF FN-Befehl definieren.

```
DEF FN ARCOT(X)=ATN(X)+PI/2
```

Die Variable PI enthält automatisch beim Einschalten des Systems den Wert 3.14159265. Aufgerufen wird diese Funktion wieder mit der Anweisung

```
FN ARCOT(X)
```

wobei X für einen eingegebenen oder berechneten Winkel steht. Das Ergebnis könnte man beispielsweise einer Variablen zuweisen, was dann so aussähe:

```
WINKEL=FN ARCOT(X)
```

Beispiel 4: Zufallszahlen innerhalb eines Bereichs errechnen

Dieses Beispiel kommt in der Praxis immer wieder vor. Sie werden dabei vor die Aufgabe gestellt, eine Zahl zwischen 2 verschiedenen Werten zu erzeugen. Die nun folgende Funktion soll Ihnen dies erleichtern.

Zunächst eine kurze Erläuterung: Die Zufallszahl sollte ganzzahlig und innerhalb eines noch festzulegenden Bereichs liegen. Im Normalfall würde man wie folgt vorgehen:

```
10    INPUT"Bereichsuntergrenze:  ",U
20    INPUT"Bereichsobergrenze:   ",O
30    Z=INT(RND(1)*(O-U)+U)
40    PRINT Z
```

Mit dem Definieren einer Benutzerfunktion sieht unser Programm dann so aus:

```
10    INPUT"Obergrenze  ",O
20    INPUT"Untergrenze ",U
30    DEF FN Z(X)=ROUND(RND(1)*(O-U)+U)
40    PRINT FN Z(X)
```

Benötigen Sie später wieder eine Zufallszahl, rufen Sie einfach mit FN Z(X) die entsprechende Funktion erneut auf.

Beispiel 5: Flächenberechnung einer Fassade

Im nächsten Beispiel steht ein Maler vor der Aufgabe, die Vorderfronten von verschieden großen Häusern zu streichen. Die Fronten haben nur eines gemeinsam: Sie haben alle eine Eingangstür und ein Fenster. Um die Anzahl der Quadratmeter zu errechnen, braucht der Maler immer eine recht lange Formel, die sich aus vielen Elementen zusammensetzt.

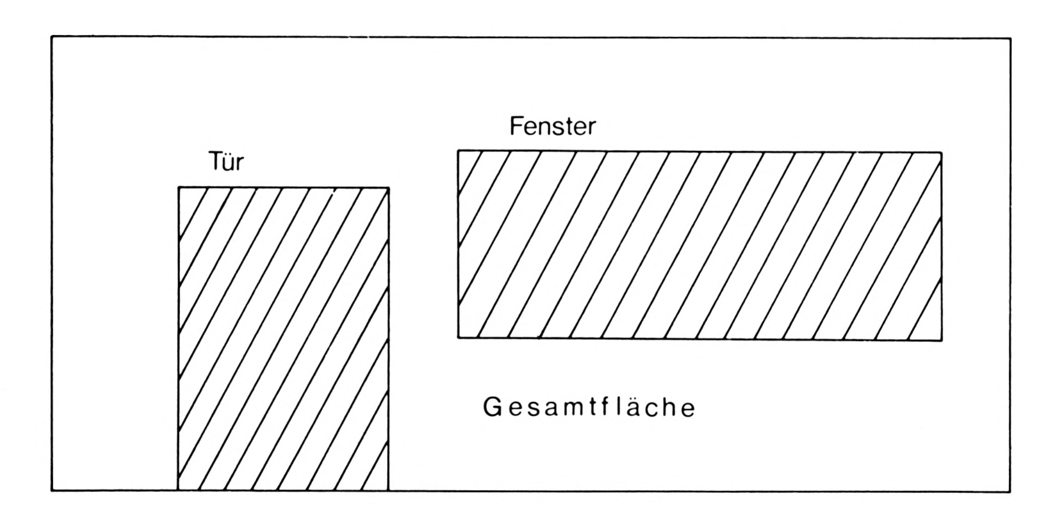


Bild 10.1: *Abbildung der Vorderfront*

Die folgenden Angaben werden zur Berechnung benötigt:

a)	Länge der gesamten Vorderfront	Variable L1
b)	Höhe der gesamten Vorderfront	Variable H1
c)	Höhe der Tür	Variable L2
d)	Breite der Tür	Variable H2
e)	Höhe des Fensters	Variable H3
f)	Breite des Fensters	Variable L3

Um die gesamte Fläche zu errechnen, die der Maler streichen muß, benötigt man folgende Formel:

```
F=L1 * H1-(L2*H2 + L3*H3)
```

Da der Maler zusätzlich für Angebote einen Preis pro Quadratmeter zugrunde legt, soll dieser im nächsten Schritt mit in die Berechnungen einbezogen werden. Es ergibt sich dann folgende Formel:

```
ANGEBOTSPREIS=F * PREIS PRO QUADRATMETER
```

Im Preis pro Quadratmeter sind das Material und die sonstigen Nebenkosten enthalten. Nun müssen die einzelnen Funktionen definiert werden. Die ersten Zeilen unseres Programms sehen dann so aus:

```
10    BORDER 0
20    MODE 2
30    INK 0,0
40    CLS
50    DEF FN FL(X)=L1*H1-(L2*H2+L3*H3)
60    DEF FN PREIS(X)= ROUND( FN FL(X)*QMP,2)
70    REM - - - - Daten eingeben - - - -
80    INPUT"Länge der Front       ";L1
90    INPUT"Höhe der Front        ";H1
100   INPUT"Höhe der Tür          ";L2
110   INPUT"Breite der Tür        ";B2
120   INPUT"Höhe des Fensters     ";H3
130   INPUT"Breite des Fensters   ";B3
140   PRINT"Die zu streichende Fläche beträgt: ";FN FL(X);" QM"
150   INPUT"Welchen Preis pro Quadratmeter nehmen ";QMP
160   PRINT"Als Angebotspreis kommt der Betrag von ";FN PREIS (X);" DM
      zustande."
```

In diesem Beispiel werden in den Zeilen 50 und 60 die beiden Benutzerfunktionen definiert, in Zeile 50 die Funktion für die Fläche und in Zeile 60 die Funktion für den Preis. Beachten Sie bitte die Kombination in Zeile 60. Hier wurde die erste Funktion mit in die Berechnung integriert. Auch solche Berechnungen sind möglich. Außerdem wird in Zeile 60 der Betrag noch auf 2 Nachkommastellen gerundet, wie es bei DM-Beträgen üblich ist.

Wird jetzt irgendwo im Programm wieder eine solche Fassade berechnet, müssen nur noch die Höhen und Breiten sowie der Quadratmeter-Preis eingegeben werden. Durch den Aufruf mit

```
PRINT FN PREIS(X)
```

erscheint dann auf Tastendruck der Angebotspreis. Einfacher geht es wirklich nicht mehr!

10.3 Was Sie noch über Benutzerfunktionen wissen sollten

Die nachfolgenden Punkte sollten Sie bei der Anwendung von selbsterstellten Funktionen beachten:

1) Die Funktion muß mit DEF definiert werden, bevor sie aufgerufen werden kann.

2) BASIC-Befehle dürfen nicht als Funktionsnamen verwendet werden. Ein unzulässiger Name wäre z.B. der Befehl TAG.

3) Beim Ausschalten gehen alle diese Funktionen wieder verloren.

4) Setzen Sie die Definitionen immer an den Anfang des Programms. Damit wird der gesamte Aufbau übersichtlicher und Fehlerquellen lassen sich schneller lokalisieren.

5) Achtung!! Der Befehl **CLEAR** löscht Benutzerfunktionen. Seien Sie deshalb vorsichtig mit der Anwendung dieses Befehls!

6) Wenn Sie oft dieselben Benutzerfunktionen in Ihren Programmen verwenden wollen, empfehle ich Ihnen, diese als Programmbausteine in Ihre Programm-Bibliothek mit aufzunehmen.

7) Arbeiten Sie möglichst häufig mit Benutzerfunktionen. Sie sparen sich damit viel Tipperei und unter Umständen auch zusätzliche Fehlerquellen.

11 Standardfunktionen

In diesem Kapitel geht es in erster Linie um die mathematischen Standardfunktionen, die der CPC 464 kennt. Viele davon sind bei komplizierten Berechnungen sehr nützlich. Im einzelnen handelt es sich um folgende Funktionen:

ABS(Z)	Absolutwert
LOG(Z)	Natürlicher Logarithmus
SQR(Z)	Quadratwurzel
EXP(Z)	Exponentialfunktion
SGN(Z)	Signumfunktion
SIN(Z)	Sinusfunktion
TAN(Z)	Tangensfunktion
COS(Z)	Cosinusfunktion
INT(Z)	Integerfunktion
FIX(Z)	Rundungsfunktion
CINT(Z)	Rundungsfunktion
ROUND(Z)	Rundungsfunktion
MIN( )	Minimalwertbestimmung
MAX( )	Maximalwertbestimmung

Interessant sind die vielen Rundungsmöglichkeiten und die Befehle MIN und MAX.

11.1 Die Funktion ABS

Diese Funktion gibt den Absolutwert einer Zahl wieder, d.h. negative Werte werden in positive umgewandelt. Dazu ein Beispiel:

```
10    INPUT ZAHL
20    X=10-ZAHL
30    PRINT"Wert normal:  ";X
40    PRINT"Absolutwert:  ";ABS(X)
```

11.2 Die Funktion LOG

Diese Funktion gibt den Wert des natürlichen Logarithmus wieder, wie im folgenden Beispiel:

```
10    MODE 1
20    CLS
30    FOR I=1000 TO 2000
40    PRINT I, LOG(I)
50    NEXT I
```

11.2.1 Die Funktion LOG10

Diese Funktion arbeitet wie die unter 11.2 beschriebene Funktion LOG, mit dem Unterschied, daß hier als Basis die 10 dient.

11.3 Die Funktion SQR

Mit dieser Funktion können die Quadratwurzeln aus beliebigen Zahlen gezogen werden. Wird versucht, aus einer negativen Zahl die Wurzel zu ziehen, gibt der Rechner eine Fehlermeldung aus:

```
10    CLS
20    FOR I=1 TO 100
30    PRINT"Die Wurzel aus ";I;" ist ";SQR(I)
40    NEXT I
```

Versuchen Sie zur Kontrolle einmal, ob sich aus der Zahl -100 die Wurzel ziehen läßt:

```
PRINT SQR(-100)
```

Jetzt sollte eine Fehlermeldung erscheinen. Eine Möglichkeit, dies zu umgehen, besteht darin, die negativen Werte mit der Funktion ABS in positive Werte umzuwandeln:

```
PRINT SQR(ABS(-100))
```

11.4 Die Funktion EXP

Hier wird die natürliche Exponentialfunktion zur Basis e berechnet, wobei e der natürliche Logarithmus von 1 ist (2.7182818). Beispiel:

```
PRINT EXP(1)
```

11.5 Die Funktion SGN

Die Funktion Signum zeigt an, um welches Vorzeichen es sich bei einer Zahl handelt. SGN gibt für negative Werte als Ergebnis -1, für positive Werte eine 1 und wenn der bearbeitete Wert Null ist, eine 0 aus. Beispiel:

```
10   FOR I=1 TO 10
20   X=INT(RND(1)*100)
30   Y=INT(RND(1)*100)
40   PRINT"Wert für X = ";X
50   PRINT"Wert für Y = ";Y
60   WERT=X-Y
70   PRINT"Als Ergebnis kommt heraus: ";WERT
80   PRINT"Die Signumfunktion liefert den Wert ";SGN(WERT)
90   PRINT"-----------------------------------------------"
100  PRINT
110  NEXT I
```

Das Programm erzeugt zwei Zufallszahlen (X und Y). Anschließend wird eine Subtraktion durchgeführt (Zeile 60), deren Ergebnis auf das Vorzeichen hin überprüft wird. Bei einer positiven Zahl erscheint eine 1, bei einer negativen Zahl eine -1. Sollte die Variable WERT den Wert 0 haben, wird als Ergebnis der Signumfunktion eine Null ausgegeben.

11.6 Die Funktion SIN

Die Funktion liefert den Sinuswert eines Winkels im Bogenmaß. Sollten Sie aber lieber in Gradmaß arbeiten, müssen Sie mit dem Befehl DEG auf Winkelgradmaß umschalten. Beispiel:

```
PRINT SIN(1)
```

11.7 Die Funktion TAN

Diese Funktion arbeitet im Prinzip wie die Sinusfunktion, nur daß hier der Tangens eines Winkels wiedergegeben wird. Beispiel:

```
PRINT TAN(1)
```

11.8 Die Funktion COS

Für die Cosinusfunktion trifft dasselbe zu, was bereits unter 11.6 und 11.7 besprochenwurde. Beispiel:

```
PRINT COS(25)
```

11.9 Die Funktion INT

Diese Funktion gibt den ganzzahligen Wert einer Zahl wieder; bei negativen Werten wird die Zahl um 1 verringert. Das folgende Beispiel soll dies verdeutlichen:

```
10    MODE 1
20    CLS
30    FOR I=1 TO 20
40    X=RND(1)*20
50    Y=RND(1)*20
60    PRINT"Wert für X =  ";X
70    PRINT"Wert für Y =  ";Y
80    WERT=X-Y
70    PRINT"Ergebnis ungerundet: ";WERT;" gerundet: ";INT(WERT)
80    PRINT STRING$(39,154)
90    NEXT I
```

Wie Sie sehen, schneidet der Befehl INT bei positiven Werten einfach die Nachkommastellen ab, rundet aber auf der anderen Seite im Minusbereich die Werte auf. So wird z.B. aus -1.999 nicht etwa der Wert -1, sondern -2.

11.10 Die Funktion FIX

Diese Funktion arbeitet ähnlich wie INT, nur daß hier im positiven wie im negativen Bereich die Nachkommastellen abgeschnitten werden. Ersetzen Sie beispielshalber dazu Zeile 70 im obigen Beispiel durch die folgende Zeile:

```
70 PRINT"Ergebnis ungerundet: ";WERT;"abgeschnitten: ";FIX(WERT)
```

11.11 Die Funktion CINT

Die Funktion CINT rundet nur auf ganzzahlige Werte, also Zahlen ohne Nachkommastellen. Das folgende Beispiel soll dies zeigen:

```
10    FOR I=1 TO 10
20    Z=RND(1)*100
30    PRINT"Wert ungerundet: ";Z;"gerundet mit CINT: ";CINT(Z)
40    PRINT
50    NEXT I
```

Wie Sie sehen werden, rundet das Programm entweder auf oder ab, je nach erzeugter Zufallszahl.

11.12 Die Rundungsfunktion ROUND

Mit ROUND können Sie dem Computer sagen, was Sie wie runden möchte. Diese Funktion läßt sich folgendermaßen anwenden:

a) ROUND(WERT)

b) ROUND(WERT, Anzahl der Nachkommastellen)

Dazu gleich ein praktisches Beispiel:

```
10    CLS
20    FOR I=1 TO 10
30    X=RND(1)*1000
40    PRINT"Zufallszahl ungerundet:  ";X; "gerundet: ";ROUND(X)
50    PRINT
60    NEXT I
```

Das Programm entspricht der unter Punkt a) aufgeführten Anwendung.

Das zweite Beispiel arbeitet mit gerundeten Werten. Gerundet bedeutet hier, daß auf mehrere Nachkommastellen gerundet werden kann. Dazu ändern Sie bitte Zeile 40 wie folgt ab:

```
40 PRINT"Zufallszahl ungerundet: ";X;"  gerundet: ";ROUND(X,2)
```

Das Programm rundet jetzt auf Nachkommastellen genau. Möchten Sie die Anzahl der Nachkommastellen frei bestimmen, geben Sie zusätzlich Zeile 5 ein und ändern das Ende der Zeile 40:

```
5 INPUT"Auf wieviele Nachkommastellen runden ";RUNDEN
```

Jetzt noch die Zeile 40 ändern. Am Schluß muß es nun heißen:

```
ROUND (X,RUNDEN)
```

Dieses Programm rundet Ihnen Ihre Werte auf die Nachkommastellen, die Sie in Zeile 5 eingegeben haben. Das Beispiel entspricht der unter Punkt b) aufgeführten Anwendung.

11.13 Die Funktion MIN

Wer stand nicht schon einmal vor dem Problem, aus einer Vielzahl von Werten die kleinste Zahl heraussuchen zu müssen. Bereits bei ungefähr 1000 Zahlen ist dies ein recht mühsames Unterfangen.

Mit der Funktion MIN ist das nun kein Problem mehr, da Sie damit den kleinsten Wert ermitteln und festhalten können. Das folgende Beispiel zeigt Ihnen die Arbeitsweise von MIN. Geben Sie dazu im Direktmodus ein:

```
PRINT MIN(100,99,67.8989,3.55,1)
```

Wenn Sie jetzt die Taste <ENTER> drücken, müßte als Ergebnis die Zahl 1 auf dem Bildschirm erscheinen.

11.14 Die Funktion MAX

Genau umgekehrt arbeitet die Funktion MAX; sie gibt den Maximalwert einer Liste aus. Geben Sie dazu das folgende Beispiel im Direktmodus ein:

```
PRINT MAX(100,200,1000,700,455,678,999)
```

Als Ergebnis dieser Funktion müßte die Zahl 1000 auf dem Bildschirm erscheinen.

12 Arbeiten mit Zufallszahlen

Für Spiele und Simulationen ist der Zufall unabdingbar. Wer möchte schon ein Videospiel spielen, bei dem er genau weiß, was als nächstes kommt?

Also wurde in jeden Computer ein sogenannter Zufallsgenerator eingebaut, der die Aufgabe hat, rein zufällige Zahlen zu erzeugen und diese zur weiteren Verarbeitung zur Verfügung zu stellen.

Wenn man von Zufallszahlen spricht, muß man mit diesem Begriff etwas vorsichtig umgehen, denn der Computer arbeitet ja nicht zufällig, sondern nach einem bestimmten Schema.

Es kann Ihnen also unter Umständen passieren, daß Sie bei den ersten Programmdurchläufen immer die gleichen Werte bekommen.

Dabei handelt es sich um eine rein technische Angelegenheit, die ich Ihnen kurz erklären möchte: Wenn der Rechner eingeschaltet wird, werden auch bestimmte Ausgangszustände angenommen. Der Rahmen und der Hintergrund werden z.B. blau und die Schrift gelb.

Diese Grundeinstellung, also einen Ausgangswert, der nach dem Einschalten immer gleich ist, nimmt auch der Zufallsgenerator an.

Um den Rechner in diesem Punkt zu überlisten, gibt es aber auch einen Befehl, den ich Ihnen gleich vorstellen möchte. Zuerst aber setzen Sie Ihren Computer in den Ursprungszustand zurück.

Der Befehl, der eine Zufallszahl erzeugt, lautet RND.

Wie der Befehl `FRE(X)` braucht auch dieser Befehl ein sogenanntes Scheinargument. Geben Sie als Versuch ein: `PRINT RND(1)`.

Sie erhalten als Ergebnis eine Zahl zwischen 0.000001 und 0.999999, auf jeden Fall einen Wert unter 1.

Natürlich ist dieser Wert für viele Anwendungen zu klein. Aber man kann ihn ja auch noch größer machen, wenn er mit einer höheren Zahl multipliziert wird.

Geben Sie im Direktmodus den Befehl PRINT RND(1)*1000 ein.

Nun wissen Sie auch, wie man größere Zufallswerte erzeugen kann.

In den nächsten Beispielen geht es um die Erzeugung von Zufallszahlen verschiedenster Größe.

Beispiel 1: Erzeugen von 10 Zufallszahlen zwischen 0.000001 und 0.999999

```
10    FOR I=1 TO 10
20    PRINT RND(1)
30    NEXT I
```

Beispiel 2: Erzeugen von 10 Zufallszahlen zwischen 0 und 99

```
10    FOR I=1 TO 10
20    PRINT RND(1)*100
30    NEXT I
```

Beispiel 3: Erzeugen von 10 gerundeten Zufallszahlen zwischen 0 und 100

```
10    FOR I=1 TO 100
20    PRINT INT(RND(1)*100)+1
30    NEXT I
```

Und nun eine Hilfe, wenn nach dem Einschalten immer wieder die selbe Zahlenfolge erscheint. Hier geht es um den zweiten Befehl im Rahmen der Zufallszahlenerzeugung. Er lautet RANDOMIZE und benötigt hinter dem reinen Befehl noch die Angabe eines numerischen Werts.

Dieser Befehl setzt den Zufallsgenerator auf einen anderen Wert, als den, den er nach dem Einschalten angenommen hat.

Ich benutze dafür immer folgende Kombination: RANDOMIZE TIME.

So erspare ich mir die Eingabe, denn die Variable für die Zeit hat mit Sicherheit nie den gleichen Wert, da sie ständig größer wird. Geben Sie RANDOMIZE so ein, werden Sie gefragt: "*Random number seed*?" Hier müßte ein Wert eingegeben werden, damit es im Programm weiter geht.

Für viele Anwendungen ist es recht sinnvoll, Zufallszahlen innerhalb eines vorbestimmten Bereichs zu erzeugen, z.B. zwischen 50 und 100. Das folgende Programm soll dies einmal demonstrieren:

```
10    MODE 1
20    CLS
30    INPUT"Bitte untere Bereichsgrenze eingeben:  ",UNTEN
40    INPUT"Bitte obere Bereichsgrenze eingeben:   ",OBEN
50    PRINT"Nun werden 10 Zufallszahlen erzeugt:"
60    FOR I=1 TO 10
70    PRINT INT(RND(1)*(OBEN-UNTEN))+UNTEN
```

```
80    NEXT I
```

In diesem Programm werden also innerhalb eines vorgegebenen Bereichs Zufallszahlen erzeugt.

Wenn Sie möchten, können Sie das Programm oben noch ein wenig ausbauen, wenn Sie folgende Änderungen vornehmen:

```
70    FARBE=INT(RND(1)*(OBEN-UNTEN))+UNTEN
73    BORDER FARBE
75    FOR PAUSE=1 TO 1000:NEXT PAUSE
```

Für Spiele bietet sich diese Möglichkeit geradezu an. Auch dazu ein kleines Beispiel:

```
10    MODE 1
20    CLS
30    INPUT"Unterer Wert    ";UNTEN
40    INPUT"Oberer Wert     ";OBEN
50    FOR I=1 TO 5
60    CLS
70    Z=INT(RND(1)*(OBEN-UNTEN))+UNTEN
80    PRINT"Bitte raten Sie eine Zahl."
90    INPUT"Was meinen Sie:  ",RATEN
100   IF RATEN>Z THEN PRINT"Leider zu groß.":PRINT:GOSUB 200:  GOTO 80
110   IF RATEN<Z THEN PRINT"Leider zu klein.":PRINT:GOSUB 200:  GOTO 80
120   IF RATEN=Z THEN BORDER 1,5:PRINT"Richtig!! Sie haben"; VERSUCH;
      " Versuche dazu gebraucht.":FOR PAUSE=1 TO 3000: NEXT PAUSE:BORDER 0
130   VERSUCH=0
140   NEXT I
150   END
200   VERSUCH=VERSUCH+1:RETURN
```

Dieses Programm ist ein Ratespiel, bei dem Sie einen Bereich eingeben müssen, in dem Sie die Zahlen raten wollen.

Danach erzeugt der Schneider seine Zufallszahl und läßt Sie raten.

Bei jedem mißglückten Versuch erhöht sich der Zähler für die Anzahl der Versuche.

Wenn die Zahl richtig war, blinkt der Rahmen ein wenig und anschließend wird die Zählvariable wieder auf 0 gesetzt und eine neue Zufallszahl erzeugt.

13 Grafik

Viele Mikrocomputer bieten heute die Möglichkeit, Informationen grafisch auf dem Bildschirm darzustellen. Dabei werden die grafischen Leistungen dieser Rechner immer besser. Der Schneider CPC 464 hat gute Grafik-Eigenschaften. Mit den zur Verfügung stehenden Grafikbefehlen können Sie ohne größeren Aufwand farbige Darstellungen selbst programmieren.

Interessant ist dabei besonders die Vielzahl der möglichen Farben, von denen insgesamt 27 zur Verfügung stehen. Leider sind sie jedoch nicht alle gleichzeitig verwendbar. Mit diesen Farben können Sie den Bildschirmrahmen, den Hintergrund und die Schriftfarbe selbst bestimmen.

Einen Nachteil hat die Arbeit mit dem Farbbildschirm allerdings, den Sie sicherlich auch bald bemerken werden: Die Augen ermüden bei längerem Arbeiten erheblich schneller, als vor dem Schwarz-Weiß-Bildschirm. Zur Arbeit vor dem Bildschirm noch einige Tips:

1. Schalten Sie neben dem Bildschirm eine kleine Lampe ein, damit Ihre Augen weniger überanstrengt werden.

2. Halten Sie einen ausreichend großen Abstand zum Bildschirm ein, d.h. mindestens 60cm oder mehr.

3. Sobald Sie bemerken, daß Ihre Augen brennen oder sogar tränen, unterbrechen Sie die Arbeit und machen Sie eine kleine Regenerationspause.

13.1 Bildschirmbetriebsarten

Für Darstellungen auf dem Bildschirm stehen Ihnen drei verschiedene Auflösungen zur Verfügung, die in nachfolgender Tabelle aufgeführt sind.

Betriebsart	Anzahl möglicher Farben	Grafikauflösung	Zeichen/Zeile
MODE 0	16	160 mal 200 Punkte	20
MODE 1	4	320 mal 200 Punkte	40
MODE 2	2	640 mal 200 Punkte	80

Wie Sie aus der Tabelle ersehen, ist es nicht möglich, alle Farben gleichzeitig darzustellen. Welche Farben Sie anwenden wollen, können Sie natürlich selbst bestimmen.

Eine Besonderheit beim CPC 464 ist, daß Sie wählen können, wieviele Zeichen in einer Zeile dargestellt werden sollen. So stehen in der Betriebsart 0 insgesamt 25 Zeilen mit je 20 Zeichen pro Zeile zur Verfügung, im Betriebsmodus 1 sind es insgesamt 25 Zeilen mit je 40 Zeichen pro Zeile und im Modus 2 sind es 80 Zeichen pro Zeile. Beim Einschalten des Rechners wird automatisch der Bildschirmmodus 1 gewählt. Diese Darstellung eignet sich besonders für längeres Arbeiten.

Ein großer Pluspunkt gegenüber Konkurrenzmodellen ist die Möglichkeit der 80-Zeichen-Darstellung. Damit können Sie auch kommerzielle Software für den CPC 464 entwickeln. Bei Verwendung des Farbbildschirms ist in der Betriebsart 2 folgende Farbkombination empfehlenswert: Rahmen- und Hintergrundfarbe auf die Farbe Schwarz setzen, Schriftfarbe auf Weiß setzen.

Als Nachteil ist zu vermerken, daß sich die drei Bildschirmbetriebsarten nicht gleichzeitig miteinander kombinieren lassen. Denkbar wäre z.B. eine Überschrift in MODE 0, eine Tabelle in MODE 2 und einige Zeilen Text in MODE 1 gewesen. Der CPC 464 löscht jedoch jedesmal, wenn er einen MODE-Befehl trifft, den gesamten Bildschirm und damit gehen alle Informationen, die vorher auf dem Bildschirm standen, verloren.

Doch nun zur detaillierten Beschreibung der einzelnen Betriebsarten.

13.1.1 MODE 0

Der Modus 0 stellt Ihnen eine Grafikauflösung von 160 mal 200 Bildpunkten zur Verfügung. Bei jedem Bildschirmpunkt können Sie wählen, welche Farbe er haben soll (max. sind 16 Farben wählbar). Diese Grafikbetriebsart kann jederzeit per Programm oder direkt von Hand durch Eingabe von MODE 0 aktiviert werden. Es wird der Bildschirm gelöscht und ein READY erscheint in der 20-Zeichen-Darstellung, wenn Sie den

Befehl MODE 0 im Kommandomodus eingeben. Probieren Sie es am besten gleich einmal aus.

Diese Darstellungsform benutze ich persönlich bei Schulungen, da man die großen Buchstaben und Zahlen auch noch aus größerer Entfernung gut ablesen kann.

13.1.2 MODE 1

Mit diesem Modus, der sich automatisch beim Einschalten des Computers aktiviert, lassen sich die meisten Anwendungen realisieren. In dieser Betriebsart können zwar nur vier verschiedene Farben gleichzeitig benutzt werden, das reicht in der Praxis meist jedoch vollkommen aus.

Sie erhalten mit Mode 1 schon eine ansehnliche Grafikauflösung von 200 mal 320 Bildschirmpunkten. Ich benutze diesen Modus immer beim Programmieren, da er recht übersichtlich alle Zeichen darstellt.

13.1.3 MODE 2

Mit dem dritten Modus, der mit MODE 2 aktiviert wird, können Sie die Darstellung von 80 Zeichen pro Zeile erreichen und erhalten eine Grafikauflösung von 640 mal 200 Bildschirmpunkten. Diese Leistung wird heute oft nur bei Mikrocomputern der gehobenen Preisklasse geboten. Bei dieser Betriebsart, in der 128000 einzelne Bildschirmpunkte angesteuert werden können, lassen sich nur zwei verschiedene Farben gleichzeitig darstellen. Dieser kleine Nachteil wird jedoch durch die hohe Auflösung behoben.

Geben Sie einmal den Befehl MODE 2 ein, und betätigen Sie die <ENTER>-Taste. Es erscheint ein kleines READY in gelber Farbe links oben am Bildschirmrand. Andere Schriftfarben lassen sich bei der Verwendung des Farbbildschirms nicht so scharf darstellen. Versuchen Sie beispielsweise einmal folgendes:

1. Schritt Schalten Sie den CPC 464 auf MODE 2 um.
2. Schritt Geben Sie den Befehl INK 1,5 ein.

Wenn Sie die <ENTER>-Taste betätigt haben, werden Sie alle Zeichen auf dem Bildschirm in der Farbe Hellviolett sehen. Wie man sieht, gibt es

auch Farben, die für die 80-Zeichen-Darstellung nicht geeignet sind, da die Schrift dann sehr unscharf wirkt.

In der nachfolgenden Übersicht finden Sie einige Farbkombinationen, die die Schrift auf dem Farbbildschirm besonders deutlich hervorheben und Ihre Augen nicht zu sehr strapazieren. Die für Sie am besten geeignete Kombination müssen Sie selbst herausfinden.

Tabelle "Günstige Farbkombinationen im 80-Zeichen-Modus":

Version	**Rahmenfarbe**	**Hintergrundfarbe**	**Schriftfarbe**
a	Schwarz	Schwarz	Hellrot (6)
b	Schwarz	Schwarz	Grün (9)
c	Schwarz	Schwarz	Blaugrün (10)
d	Schwarz	Schwarz	Gelb (12)
e	Schwarz	Schwarz	Weiß (13)
f	Schwarz	Schwarz	Orange (15)
g	Schwarz	Schwarz	Limonengrün (21)
h	Schwarz	Schwarz	Pastellgrün (22)

Wie Sie aus der Tabelle entnehmen können, sind die Rahmen- und Hintergrundfarben immer dunkel gehalten worden, wodurch ein guter Kontrast zum Schriftbild erreicht wird. Regeln Sie nach Möglichkeit auch die Helligkeit des Bildschirms solange, bis Sie das gewünschte Schriftbild erreicht haben.

Das beste Schriftbild im 80-Zeichen-Modus erhalten Sie, wenn Sie den monochromen Bildschirm verwenden, der nur verschiedene Grünschattierungen darstellen kann. Damit können Sie auch längere Zeit vor dem Computer arbeiten.

Das folgende Programm stellt Ihnen die Möglichkeiten der verschiedenen Darstellungen vor. Es enthält einige Befehle, die Sie bisher noch nicht kennengelernt haben. Diese Befehle ändern die Farben von Bildschirmrahmen, Hintergrund und Schrift und werden nach diesem Demonstrationsprogramm ausführlich besprochen.

```
10    border 0:rem  Bildschirmrahmen auf Schwarz setzen
20    mode 0:  rem  20-Zeichen-Modus anwählen
30    for i=1 to 13
40    ink 1,j: rem  Farbspeicher 1 mit der i.-ten Farbe füllen
50    paper i: rem  Hintergrundfarbe setzen
60    pen i+1: rem  Zeichenfarbe setzen
70    cls:     rem  Bildschirm komplett löschen
80    print"Das ist der Modus 0"
90    print"==================="
100   print
110   print"Damit kann man 20"
```

```
120   print
130   print"Zeichen in einer "
140   print"Zeile darstellen !"
150   print
160   print
170   print
180   print
190   print"Bitte eine Taste"
200   print"druecken........"
210   a$=inkey$:if a$="" then 210:rem  Tastaturabfrage
220   next:rem Schleife für die 20-Zeichen-Darstellung schließen
230   for i=1 to 5000:next:rem  Warteschleife
240   mode 1:rem  40-Zeichen-Modus anwählen
250   ink 0,0:rem  Farbspeicher 0 mit der Farbe Schwarz füllen
260   pen 1:  rem  Schriftfarbe aus Farbspeicher 1 holen
270   paper 0:rem  Hintergrundfarbe aus Farbspeicher 0 holen
280   cls:    rem  Bildschirm löschen
290   for i=0 to 26 :rem  alle Farben einmal durchlaufen
300   ink 1,i: rem        in Farbspeicher 1 die Farbe i setzen
310   cls:     rem        Bildschirm löschen
320   print"Das ist nun der Modus 1, der gleich beim";
330   print
340   print"Einschalten des Rechners aktiviert ist."
350   print
360   print"In diesem Modus koennen nur noch 4 Far-"
370   print
380   print"ben fuer Schrift und Bildschirmhinter-"
390   print
400   print"grund gleichzeitig verwendet werden."
410   print
420   print
430   print"Verwendete Farbnummer : ";i
440   a$=inkey$:if a$="" then 440:rem Tastaturabfrage
450   next: rem  Schleife für 40-Zeichen-Darstellung schließen
460   for i=1 to 2000:next:rem  Warteschleife durchlaufen
470   cls:rem  Bildschirm löschen
480   print"Als nächstes wird die 80-Zeichen-Dar-"
490   print
500   print"stellung demonstriert, mit der auch"
510   print
520   print"kommerzielle Anwendungen möglich sind."
530   for i=1 to 2000:next:cls:rem Warteschleife durchlaufen und Bildschirm
      komplett löschen
540   mode 2:rem  80-Zeichen-Darstellung anwählen
550   ink 0,0:rem  Farbspeicher 0 mit der Farbe Schwarz füllen
560   paper 0:rem  Hintergrundfarbe aus Farbspeicher 0 holen
570   for i=0 to 26: rem  alle Farben einmal durchlaufen
580   ink 1,i:rem  Farbspeicher 1 mit der i.-ten Farbe füllen
590   pen 1:  rem  Schriftfarbe aus Farbspeicher 1 holen
600   print"Das ist der Modus 2 fuer die Darstellung von bis zu 80 Zeichen pro
      Zeile."
610   print
620   print"Die beste Auflösung erhalten Sie bei Verwendung des gruenen
      Monitors."
630   print
640   print
650   print" Verwendete Farbnummer : ";i
```

```
660   a$=inkey$:if a$="" then 660
670   cls:rem   Bildschirm löschen
680   next:rem  Schleife schließen
690   cls:rem   Bildschirm löschen
700   locate 20,10:rem  Textcursor neu positionieren
710   ink 1,3,9:rem  in Farbspeicher 1 werden 2 Farben gesetzt
720   pen 1:rem aus Farbspeicher 1 beide Farben für die Schrift holen
730   print"Dieses Demoprogramm ist an dieser Stelle zu Ende."
740   for i=1 to 3000:next:rem  Warteschleife durchlaufen
750   ink 1,3: rem  in Farbspeicher 1 die Farbe Rot setzen
760   pen 1:rem     Zeichenfarbe aus Farbspeicher 1 holen
770   list:rem      Programm wird am Ende komplett aufgelistet
```

13.2 Die Farben

Wie Sie sicherlich schon bemerkt haben, erscheint beim Einschalten des CPC 464 die Schrift immer in Gelb, während Bildschirmrahmen und Hintergrund immer Blau sind.

Dies sind natürlich nicht die einzigen Farben, die der Schneider darstellen kann. Die folgende Tabelle gibt Ihnen einen Überblick, welche Farben zum Arbeiten zur Verfügung stehen.

Farbnr.	Farbe	Farbnr.	Farbe
0	Schwarz	17	Pastellmagenta
1	Blau	18	Hellgrün
2	Hellblau	19	Seegrün
3	Rot	20	Hellblaugrün
4	Magenta	21	Limonengrün
5	Hellviolett	22	Pastellgrün
6	Hellrot	23	Pastellblaugrün
7	Purpur	24	Hellgelb
8	Magentahell	25	Pastellgelb
9	Grün	26	Hellweiß
10	Blaugrün	27	Weiß
11	Himmelblau	28	Purpur
12	Gelb	29	Pastellgelb
13	Weiß	30	Blau
14	Pastellblau	31	Seegrün
15	Orange		
16	Rosa		

Teilweise finden Sie in der Farbtabelle auch einige "exotische" Farben, die sich zwar vom Namen her gut anhören, auf dem Bildschirm aber nicht sehr gut wirken. Dies betrifft in erster Linie Farbkombinationen wie Magentahell oder Hellviolett. Wie anfangs bereits erwähnt, lassen sich nicht alle Farben gleichzeitig auf dem Bildschirm darstellen. In Modus 0 waren

es 16, im Normalmodus vier und im Modus 2 nur noch zwei Farben gleichzeitig.

Allerdings gibt es auch hier eine Ausnahme, die ich im nächsten Abschnitt behandeln möchte.

13.2.1 Die Rahmenfarbe

Die Rahmenfarbe kann beliebig gewählt werden, gleichgültig welchen Bildschirmmodus Sie gerade gewählt haben. Sie können also jede der 31 Farbnummern für den Rahmen benutzen. Der Befehl dafür lautet:

```
BORDER Farbnummer
```

Dazu gleich ein Beispiel. Versetzen Sie den Computer in den Anfangszustand (<SHIFT>, <CTRL> und <ESC> gleichzeitig drücken), und geben Sie danach den Befehl BORDER 0 ein. Wenn Sie jetzt die <ENTER>-Taste betätigt haben, hat sich der Bildschirmrahmen von Blau in Schwarz verfärbt. Nehmen Sie sich als nächstes die Farbtabelle zur Hand und versuchen Sie es auch einmal mit anderen Farbnummern.

Wir wollen nun einen Schritt weitergehen und den CPC 464 so programmieren, daß er ohne unser Eingreifen alle möglichen Rahmenfarben darstellt. Dies läßt sich mit Hilfe einer Schleife realisieren.

```
10    for i=0 to 31
20    border i
30    for p=1 to 2000:next p:rem   Warteschleife durchlaufen
40    next i
```

Wandeln Sie nun das Programm so ab, daß auch die jeweilige Nummer der Farbe auf dem Bildschirm ausgedruckt wird. Zusätzlich können Sie die Warteschleife in Zeile 30 durch eine Tastaturabfrage ersetzen. Dann wird jedesmal, wenn eine Taste gedrückt wird, eine andere Rahmenfarbe dargestellt.

Vielleicht wundern Sie sich darüber, daß es statt 26 Farbnummern plötzlich 31 Farbnummern gibt? Durch Zufall stieß ich beim Experimentieren auf diese Besonderheit und stellte fest, daß die nachfolgend aufgeführten Farben doppelt in der Tabelle enthalten sind:

Farbbezeichnung	Nummer
Weiß	27
Purpur	28
Pastellgelb	29
Blau	30
Seegrün	31

Wenn Sie ausprobieren möchten, ob es nach der Farbnummer 31 noch weitere Farben gibt, ändern Sie den Endwert der Schleife in Zeile 10 auf 35 ab. Sobald das Programm bei der letzten Farbnummer (31) angekommen ist, meldet sich der Rechner mit einer Fehlermeldung und bricht das Programm anschließend ab.

Für bestimmte Effekte ist es manchmal ganz sinnvoll, den Bildschirmrahmen zwischen 2 Farben blinken zu lassen, was Sie mit folgendem Programm erreichen können:

```
10    for i=1 to 20:rem Schleife 20 mal durchlaufen
20    border 0:for p=1 to 500:next p:rem Rahmenfarbe auf Schwarz setzen und
      halten
30    border 1:for p=1 to 500:next p:rem Rahmenfarbe auf Blau setzen und
      halten
40    next i
```

Die Geschwindigkeit, mit der zwischen den beiden Farben hin- und hergeschaltet wird, läßt sich durch Ändern der Endwerte in den beiden Warteschleifen bestimmen. Das Programm hat jedoch einen kleinen Nachteil: Während der Bildschirmrahmen zwischen den beiden Farben wechselt, kann das Programm keine anderen Tätigkeiten ausführen.

Der CPC 464 stellt fürs Blinken des Bildschirmrahmen seinen Befehl zur Verfügung. Er lautet:

```
BORDER Farbnummer A, Farbnummer B
```

Für die Nummern A und B werden die gewünschten Farbnummern eingesetzt und schon blinkt der Bildschirmrahmen. Dazu gleich ein kurzes Programm:

```
10    input"Farbnummer 1 ";a
20    input"Farbnummer " ";b
30    border a,b
40    print"Das Programm macht unabhängig vom Blinken weiter."
50    new
```

Das Programm fragt Sie nach der ersten und der zweiten Farbnummer. Sobald diese Werte eingegeben worden sind, aktiviert es in Zeile 30 das Blinken. Anschließend wird noch ein Text ausgegeben und das Programm danach gelöscht. Obwohl sich jetzt kein Programm mehr im Speicher

befindet, blinkt der Bildschirmrahmen noch immer. Sie können dies beenden, indem Sie erneut einen BORDER-Befehl, gefolgt von einer Farbnummer, eingeben.

Selbstverständlich können Sie auch die Blinkgeschwindigkeit bestimmen. Dafür stellt Ihnen der Schneider den SPEED-Befehl zur Verfügung. Mit diesem Befehl sagen Sie dem Computer, wie lang eine der beiden Farben gehalten werden soll. Der SPEED-Befehl hat folgendes Format:

```
SPEED INK Zeit für Farbnummer A, Zeit für
Farbnummer B
```

Geben Sie dazu einmal das folgende Beispiel ein:

```
10    input"Farbnummer 1 ";a
20    input"Farbnummer 2 ";b
30    border a,b
40    speed ink 50,100
```

Bis Zeile 30 ist Ihnen das Programm bereits bekannt. In Zeile 40 wird die Zeit bestimmt, wie lang welche Farbe dargestellt werden soll. Der Wert 60 in Zeile 40 bestimmt, daß die Farbe A genau eine Sekunde gehalten wird. Der zweite Wert, die Zahl 100, hält die Farbe B 2 Sekunden lang. Diese Zeiten ermittelt man, indem man die Angaben für A (Wert=50) und B (Wert=100) mit dem Faktor 0.02 multipliziert.

Ein anderes Beispiel:

```
10    border 0,1
20    speed ink 250,50
```

Für die Farbe Schwarz (0) ergibt sich folgende Zeit: 250*0.02=5 sec. Für die Farbe Blau (1) ergibt sich folgende Zeit: 50*0.02=1 sec. Die Farbe Blau erscheint also 1 Sekunde lang, während Schwarz rund 5 Sekunden dargestellt wird.

Bitte beachten Sie bei Ihren Programmentwicklungen, daß ein schneller Wechsel zwischen hellen und dunklen Farben zu Störungen des Nervensystems führen kann.

13.2.2 Die Farbspeicher

Beim Einschalten des Schneiders wird, wie schon erwähnt, eine bestimmte Farbkombination für Rahmen-, Zeichen- und Schriftfarbe gewählt. Während das Ändern der Farben für den Bildschirmrahmen keine Pro-

bleme bereitet, stößt man bei Zeichen- und Hintergrundfarbe auf einige Schwierigkeiten.

Mit den Befehlen **PEN** und **PAPER** haben Sie die Möglichkeit, die Farben für Schrift und Hintergrund frei zu wählen. Anfangs fiel mir die Benutzung dieser Befehle nicht ganz leicht, bis ich schließlich hinter die Arbeitsweise der Farbeinstellung kam.

Der CPC 464 stellt in 16 verschiedenen Farbspeichern bestimmte Farben auf Abruf zur Verfügung. Das muß man sich wie folgt vorstellen:

1. Im Farbspeicher Nummer 0 liegt standardmäßig die Farbe Blau auf Abruf bereit.

2. Möchte man auch die Schrift in Blau haben, wird mit dem Befehl **PEN 0** aus Farbspeicher 0 die Farbe Blau geholt und die Schriftfarbe gesetzt.

3. Soll beispielsweise die Schrift die Farbe Hellrot erhalten, muß zuerst geprüft werden, in welchem Farbspeicher diese Farbe liegt.

Standardmäßig liegt sie in Farbspeicher 7.

Mit dem Befehl **PEN 7** holen wir also die Farbe Hellrot aus dem Farbspeicher und färben damit die Zeichen ein.

Dieses Prinzip funktioniert natürlich auch bei Färbung des Bildschirmhintergrundes in eine der 26 verschiedenen Farben. Dafür steht der Befehl **PAPER** zur Verfügung. Welche Farben standardmäßig in welchem Farbspeicher stehen, ersehen Sie aus folgender Tabelle.

Tabelle "Standardmäßige Farbspeicherbelegung"

Farbspeicher-Nummer	MODE 0	MODE 1	MODE 2
0	Blau	Blau	Blau
1	Hellgelb	Hellgelb	Hellgelb
2	Hellblaugrün	Hellblaugrün	Blau
3	Hellrot	Hellrot	Hellgelb
4	Leuchtendweiß	Blau	Blau
5	Schwarz	Gelb	Hellgelb
6	Hellblau	Hellblaugrün	Blau
7	Magentahell	Hellrot	Hellgelb
8	Blaugrün	Blau	Blau
9	Gelb	Hellgelb	Hellgelb
10	Pastellblau	Hellblaugrün	Blau
11	Pastellblau	Hellrot	Hellgelb
12	Hellgrün	Blau	Blau
13	Pastellgrün	Hellgelb	Hellgelb
14	Blau, Hellgelb	Hellblaugrün	Blau
15	Rosa, Himmelblau	Hellrot	Hellgelb

13.2.3 Die Zeichenfarbe

Oft kommt es vor, daß man bestimmte Texte oder Buchstaben in zwei verschiedenen Farben darstellen möchte. Um dies zu realisieren, benutzt man den Befehl PEN.

Dazu gleich ein kleines Beispiel, bei dem mehrere Schriftfarben gleichzeitig auf dem Bildschirm dargestellt werden sollen.

```
10    cls:rem Bildschirm löschen
20    print" F a r b t e s t "
30    print" ================"
40    print
50    pen 3:rem Farbe aus Farbspeicher 3 holen (Hellrot)
60    print"Dieser Text erscheint in der Farbe Hellrot."
70    pen 2:rem     Farbe aus Farbspeicher 2 holen (Helltürkis)
80    print"Dieser Text erscheint in Helltürkis."
90    pen 1:rem Farbe aus Farbspeicher 1 holen (Hellgelb)
100   print"Diese Schrift erscheint wieder in Gelb."
```

Wie Sie sehen, muß nach dem Befehl PEN eine der 16 Nummern für den Farbspeicher stehen, der angesprochen werden soll. Möglich sind dabei Werte, die zwischen 0 und 15 liegen. Soll ein Farbspeicher angesprochen werden, der höher liegt als 15, erscheint die Fehlermeldung "*Improper argument*", da es nur 16 Farbspeicher gibt. Der Computer fängt mit seiner Zählung nicht bei 1, sondern bei der Zahl 0 an!

Das oben gezeigte Beispiel wurde für den Modus 1, den Standardmodus, geschrieben. Als nächstes möchte ich Ihnen ein Programm vorstellen, das im Modus 0 arbeitet. Hier stehen standardmäßig einige Farben mehr zur Verfügung.

Aufgabe:

Es sollen im Modus 0 mehrere Schriftfarben gleichzeitig auf dem Bildschirm erscheinen.

Verarbeitung:

Im Modus 0 wird mit Hilfe einer Schleife jeder Farbspeicher einmal durchlaufen und dabei ein Text ausgedruckt.

```
10    mode 0:rem Umschalten auf Modus 0
20    cls:rem Bildschirm löschen
30    for i=0 to 15:rem Schleife 16 mal durchlaufen
40    pen i:rem Jedesmal andere Schriftfarbe wählen
50    print"Farbe,Farbe,Farbe.."
60    next i:rem Schleife schließen
```

Wenn Sie das Programm eingegeben und gestartet haben, sehen Sie jetzt verschiedene Farben auf dem Bildschirm. Besonders interessant sind dabei die Farbspeicher 14 und 15. Sie enthalten nicht eine, sondern gleich zwei Farben.

Sehen Sie sich noch einmal die Tabelle für die Standardfarbbelegung an. Im Farbspeicher 14 stehen die Farben Blau und Hellgelb. Wird dieser Farbspeicher mit PEN 14 angesprochen, dann wechselt die Farbe des nachfolgenden Textes immer zwischen diesen beiden Farben ab. In unserem Beispiel sieht man allerdings nur die Farbe Hellgelb, da der Hintergrund Blau ist.

Genau wie in Farbspeicher 14, stehen auch in 15 zwei verschiedene Farben, Rosa und Himmelblau. Wird mit PEN 15 dieser Farbspeicher angesprochen, wechselt der CPC 464 immer zwischen diesen beiden Farben hin und her.

Diesen Farbwechsel können Sie nur mit dem Befehl PEN, gefolgt von einer Farbnummer bis 13, wieder aufheben. Die Zeichenfarbe wird dann wieder geändert und Sie können normal weiterarbeiten. In der Standardfarbtabelle haben Sie sicherlich bemerkt, daß im Modus 2 immer wieder die gleichen Farben auftauchen, nämlich Blau und Hellgelb. Geben Sie einmal folgendes Beispiel ein:

```
10    mode 2:rem Modus 2 anwählen
20    pen 1:rem Zeichenfarbe aus Farbspeicher 1 holen
30    print"Das ist der erste Text."
```

```
40    pen 2:rem Zeichenfarbe aus Farbspeicher 2 holen
50    print"Das ist der zweite Text."
60    print
70    print
```

Programmerklärung:

Sobald das Programm gestartet wurde, werden Sie Zeile 50 vermissen und das READY nach Programmende! Dies hängt damit zusammen, daß in Zeile 40 die Schriftfarbe neu gewählt wird und dann identisch ist mit dem Bildschirmhintergrund. Deshalb sehen Sie den Text in Zeile 50 nicht. Wenn Sie das Programm um die nachfolgenden Zeilennummern ergänzen, können Sie sich davon selbst überzeugen.

Programmergänzung:

```
80    pen 1:rem  Zeichenfarbe wieder aus Farbspeicher1 holen
90    print"Programmende"
```

Wie Sie auch mit den anderen Farbtönen arbeiten, werden wir im nächsten Abschnitt erläutern.

13.2.4 Ändern der Standardfarben

Bis jetzt wurde nur mit denjenigen Farben gearbeitet, die beim Einschalten des Schneider CPC 464 in den Farbspeichern waren. Nun wollen wir unsere Farbtöne einmal frei wählen. Dazu stellt der Computer den BASIC-Befehl INK zur Verfügung. Dieser Befehl hat folgendes Format:

```
INK Farbspeichernr., neue Farbnr.
```

Ein kleines theoretisches Beispiel dazu: `INK 1,17` ändert die Farbe in Farbspeicher 1 von ursprünglich Hellgelb (24) in die Farbe Pastellmagenta (17). Wird jetzt mit `PEN 1` diese Farbe aus dem Farbspeicher 1 geholt, erscheint alles, was als nächstes mit `PRINT` ausgegeben wird, in der Farbe Pastellmagenta. Ein Beispiel zum Nachvollziehen:

```
10    mode 1
20    ink 1,17:rem in Farbspeicher 1 die Farbe Pastellmagenta setzen
30    pen 1:rem Aktuelle Zeichenfarbe aus Farbspeicher 1 holen
40    print"Verwendete Farbe : Pastellmagenta"
50    print
60    ink 2,28:rem in Farbspeicher 2 die Farbe Purpur setzen
70    pen 2:rem Aktuelle Zeichenfarbe aus Farbspeicher 2 holen
80    print"Verwendete Farbe : Purpur"
90    print
100   ink 3,0:rem in Farbspeicher 3 die Farbe Schwarz setzen
```

```
110   pen 3:rem Aktuelle Farbe aus Farbspeicher 3 holen
120   print"Verwendete Farbe: Schwarz"
```

Dieses Programm bringt in die Farbspeicher 1 bis 3 jeweils neue Farben, die dort standardmäßig nicht vorhanden sind. Mit PEN 1, PEN 2 und PEN 3 werden diese Farben jedesmal aufgerufen und die nachfolgende Bildschirmausgabe entsprechend gefärbt.

13.2.5 Die Farbe des Bildschirmhintergrundes

Für die Einstellung des Bildschirmhintergrundes hält der CPC 464 den BASIC-Befehl PAPER bereit. Mit diesem Befehl kann man wählen, welche der 26 möglichen Farben als Hintergrund verwendet werden soll. Der PAPER-Befehl hat folgendes Format:

```
PAPER Farbspeicher
```

Bei der Angabe des Farbspeichers können Sie wieder einen Farbspeicher zwischen 0 und 15 nehmen. Probieren Sie einmal folgendes aus:

```
10    mode 1:rem Modus 1 einschalten
20    for i=0 to 15:rem Schleife 16 mal durchlaufen
30    ink 1,i:rem Farbspeicher 1 mit der Farbe i füllen
40    paper i:rem Hintergrund teilweise mit Farbe i füllen
50    cls:rem Hintergrund löschen und komplett mit neuer Farbe einfärben
60    ink 2,24:rem Farbspeicher 2 mit Hellgelb füllen
70    pen 2:rem Schriftfarbe aus Farbspeicher 2 holen
80    print"Bitte drücken Sie eine beliebige Taste...."
90    a$=inkey$:if a$="" then 90:rem Tastaturabfrage
100   next i:rem Schleife wieder schließen
```

Das Programm durchläuft alle im Modus 1 enthaltenen Standardfarben und färbt mit diesen Farben den Hintergrund ein. Auf Tastendruck erscheint eine neue Hintergrundfarbe und zwar so lange, wie die Schleife durchlaufen wird. Probieren Sie es ruhig einmal mit anderen Farben aus. Sie können dabei auch 2 verschiedene Farben in einen Farbspeicher bringen und dann den Bildschirmhintergrund ständig die Farbe wechseln lassen. Beispiel:

```
INK 1,5,6
```

bewirkt, daß der Hintergrund ständig zwischen den beiden Farben Hellviolett und Hellrot wechselt.

Versuchen Sie doch einmal, das oben aufgeführte Programm zu erweitern und zwar so, daß insgesamt 4 Schriftfarben zu sehen sind. Sicherlich werden Sie dabei auf einige Schwierigkeiten stoßen.

Wie anfangs schon erwähnt, kann der Schneider CPC 464 insgesamt vier Farben gleichzeitig im Modus 1 darstellen. Dabei stehen drei Farben für die Schrift zur Verfügung und eine für den Hintergrund! Es ist demgemäß nicht möglich, im Modus 1 vier verschiedene Schriftfarben darzustellen.

Möchte man dennoch mehr als drei Schriftfarben darstellen, kann man sich eines kleinen Tricks bedienen. Sie erinnern sich, daß in der Standardfarbtabelle im Modus 0 die Farbspeicher 14 und 15 mit je zwei verschiedenen Farben belegt waren. Dies ist auch im Modus 1 möglich! Erweitern Sie dazu das letzte Programm um die folgenden Zeilennummern:

```
20    ink 1,17,3
60    ink 2,28,29
100   ink 3.0.5
```

Der Inhalt eines jeden Farbspeichers wird um jeweils eine Farbe vergrößert. Werden nun die einzelnen Farbspeicher mit PEN 1, PEN 2 und PEN 3 angesprochen, dann blinkt jeder Text auch noch zwischen zwei verschiedenen Farben hin und her.

Mit diesem Befehl haben Sie also die Möglichkeit, einige neue Effekte in Ihre Programme einzubauen. Denkbar ist z.B. die Ausgabe einer Fehlermeldung, die zwischen zwei Farben abwechselt. Ihrer Phantasie sind nunmehr keine Grenzen gesetzt.

13.3 Der Grafikbefehl PLOT

Mit dem PLOT-Befehl ist es jetzt möglich, Linien, Kreise und andere Figuren auf dem Bildschirm zu zeichnen. Dabei wird die darzustellende Grafik aus lauter Einzelpunkten zusammengesetzt. Dieser Befehl hat folgendes Format:

```
PLOT X, Y, Farbspeichernummer
```

Zwingend notwendig sind die Angaben für die X- und Y-Achse, während die Angabe des gewünschten Farbspeichers auch entfallen kann. Dazu ein Beispiel, damit Sie sich die Wirkungsweise dieses Befehls vor Augen führen können.

```
10    mode 1
20    cls
30    plot 200,200
```

Starten Sie das Programm mit RUN und Sie werden sehen, daß an der Position 200 der X-Achse und der Y-Achse ein kleiner Punkt erscheint. Die Farbe des dargestellten Punktes ist gelb. Wenn Sie aber eine andere Farbe bevorzugen, z.B. die Farbe Rot, setzen Sie diese in den Farbspeicher 2. Da Rot die Farbnummer 6 hat, lautet der Befehl dazu `INK 2,6`. Erweitern Sie bitte Ihr Programm um folgende Zeilennummern:

```
40    ink 2,6
50    plot 300,205,2
```

Diese Programmerweiterung erzeugt einen roten Punkt an den angegebenen Koordinaten. Experimentieren Sie ruhig auch mit anderen Koordinatenangaben und anderen Farben.

Nun wollen wir noch einen Schritt weitergehen. Es soll vom Computer eine Linie gezogen werden, die sich aus lauter Einzelpunkten zusammensetzt.

Für solche Aufgaben bietet sich eine Schleife als Wiederholungsanweisung geradezu an. Sie müssen dabei nur angeben, wieviele Punkte nacheinander gesetzt werden sollen. Geben Sie einmal das folgende Programm ein:

```
10    mode 1
20    cls
30    move 0,0
40    for i=1 to 200
50    plot i,1
60    next i
```

Programmbeschreibung:

Zeile 10:	Modus 1 wird angewählt
Zeile 20:	der Bildschirm wird komplett gelöscht
Zeile 30:	der Grafikcursor wird in Position 0 gesetzt
Zeile 40:	eine Schleife wird 200mal durchlaufen
Zeile 50:	bei jedem neuen Durchlauf wird der neue Einzelpunkt eine Position weiter gesetzt
Zeile 60:	die geöffnete Schleife wird wieder geschlossen

Das Programm erzeugt eine Linie, welche unten auf dem Bildschirm dargestellt wird. Verantwortlich dafür ist der PLOT-Befehl in Zeile 50. Ändern Sie doch einfach einmal Zeile 50 und schreiben Sie statt `PLOT I,1` den Befehl `PLOT 1,I`. Wenn das Programm erneut gestartet wird, sehen Sie eine Linie, die sich von unten nach oben aufbaut.

Bis jetzt haben wir nur Linien erzeugt, die die Standardfarbe Gelb hatten. Im nächsten Programm sehen Sie, wie farbige Linien entstehen.

```
10    mode 1
20    cls
30    move 0,0
40    for i=1 to 250
50    plot i,0,1
60    plot 0,i,2
70    next
```

Nachfolgend stelle ich Ihnen einige Beispiele vor, die mit dem Befehl PLOT farbige Darstellungen auf dem Bildschirm ausgeben.

Beispiel 1: 4000 Punkte

Aufgabe:

Es sollen 4000 Punkte auf den Bildschirm gebracht werden, bei denen für die Positionen X und Y die Werte per Zufall bestimmt werden. Dabei sollen verschiedene Farben zum Einsatz kommen.

Verarbeitung:

Über den Zufallsgenerator den X- und Y-Wert, sowie die Zeichenfarbe wählen.

Ausgabe:

4000 Punkte, die sich an unterschiedlichen Positionen befinden.

Das Programm:

```
10    ink 1,13
20    ink 2,24
30    ink 3,6
40    ink 4,7
50    ink 0,0
60    mode 1
70    for i=1 to 4000
80    zf=int(rnd(1)*3)+1
90    x=int(rnd(1)*640)
100   y=int(rnd(1)*400)
110   plot x,y,zf
120   next i
```

Programmbeschreibung:

Zeile 10:	Farbnummer 13 in den Farbspeicher 1 legen
Zeile 20:	Farbnummer 24 in den Farbspeicher 2 legen
Zeile 30:	Farbnummer 6 in den Farbspeicher 3 legen
Zeile 40:	Farbnummer 7 in den Farbspeicher 4 legen
Zeile 50:	Farbnummer 0 in den Farbspeicher 0 legen

Zeile 60: Bildschirmbetriebsart 1 aktivieren
Zeile 70: Schleife öffnen
Zeile 80: Zeichenfarbe berechnen
Zeile 90: Wert für die X-Position berechnen
Zeile 100: Wert für die Y-Position berechnen
Zeile 110: an den berechneten Koordinaten einen farbigen Punkt setzen
Zeile 120: Schleife schließen

Als Abwandlung dieses Programms kann man auch den Bereich einengen, in dem die Einzelpunkte erscheinen sollen. Dazu braucht man nur die Werte in den Zeilen 90 und 100 entsprechend zu verändern. Versuchen Sie es einmal mit höheren und niedrigeren Werten.

Beispiel 2: Sieben farbige Linien

Aufgabe:

Es sollen 7 vertikale Linien mit dem PLOT-Befehl erzeugt werden, die bei der Position 0 beginnen und bis zur Position 399 gehen. Die Linien sollen farbig sein und einen gleichmäßigen Abstand zueinander haben.

Verarbeitung:

Innerhalb einer Schleife auf den Positionen 0, 100, 200, 300, 400, 500 und 600 einen Einzelpunkt setzen.

Ausgabe:

Sieben farbige Linien

Das Programm:

```
10    mode 1
20    cls
30    input"Welche Schrittweite ? (max. 10)  ", schrittweite
40    cls
50    move 0,0
60    for i=0 to 399 step schrittweite
70    plot 0,i,1
80    plot 100,i,2
90    plot 200,i,3
100   plot 300,i,2
110   plot 400,i,3
120   plot 500,i,5
130   plot 600,i,6
140   next i
150   taste$=inkey$
```

```
160   if taste$="" then goto 150
170   run
```

Programmbeschreibung:

Zeile 10: Umschalten in Betriebsart 1
Zeile 20: Bildschirm löschen
Zeile 30: Schrittweite eintragen
Zeile 40: Bildschirm löschen
Zeile 50: Grafikcursor in Ausgangsposition bringen
Zeile 60: Schleife öffnen
Zeile 70: auf die Position 0 einen farbigen Punkt setzen
Zeile 80: auf die Position 100 einen farbigen Punkt setzen
Zeile 90: auf die Position 200 einen farbigen Punkt setzen
Zeile 100: auf die Position 300 einen farbigen Punkt setzen"
Zeile 110: auf die Position 400 einen farbigen Punkt setzen"
Zeile 120: auf die Position 500 einen farbigen Punkt setzen"
Zeile 130: auf die Position 600 einen farbigen Punkt setzen"
Zeile 140: Schleife wieder schließen
Zeile 150: Tastaturabfrage
Zeile 160: Wenn keine Taste gedrückt, dann wieder nach Zeile 150 gehen
Zeile 170: Programmstart

Beispiel 3: Koordinatenkreuz

Aufgabe:

Koordinatenkreuz zeichnen

Verarbeitung:

Mit Hilfe einer Schleife und dem PLOT-Befehl durch die Bildschirmmitte 2 Linien zeichnen.

Ausgabe:

2 Linien in der Mitte des Bildschirms

Das Programm:

```
10    mode 1
20    cls
30    move 0,0
40    for i=0 to 399
50    plot 320,i
60    next i
70    move 0,0
```

```
80    for i=0 to 639
90    plot i,200
100   next i
```

Programmbeschreibung:

Zeile 10: Betriebsmodus 1 wählen
Zeile 20: Bildschirm löschen
Zeile 30: Grafikcursor in die Ausgangsposition setzen
Zeile 40: Schleife öffnen für die Y-Achse
Zeile 50: auf Position 320 einen Punkt setzen
Zeile 60: Schleife wieder schließen
Zeile 70: Grafikcursor in Ausgangsposition setzen
Zeile 80: Schleife öffnen für die X-Achse
Zeile 90: auf die Position 200 einen Punkt setzen
Zeile 100: Schleife wieder schließen

Beispiel 4: Spirale

Aufgabe:

Auf dem Bildschirm soll eine Spirale geplottet werden.

Verarbeitung:

Mit Hilfe eines immer kleiner werdenden Durchmessers wird eine Spirale gezeichnet. Ist die Bildschirmmitte (320,200) erreicht, soll das Programm beendet werden. Das Aussehen der Spirale kann mit Hilfe der Schrittweite variiert werden. Die aktuellen Grafikcursorpositionen werden rechts auf dem Bildschirm ausgegeben.

Ausgabe:

Spirale

Das Programm:

```
10    mode 1
20    input"Schrittweite  : ",x
30    cls
40    ink 2,6
50    move 320,200
60    d=190
70    locate 24,1:print"Aktuelle Cursor-"
80    locate 30,2:print"position : "
90    deg
100   for i=0 to 100000 step x
110   rem  Textcursor positionieren
120   locate 32,3:pen 2:print"x= ";xpos
```

```
130  locate 32,4:print"y = ";ypos
140  d=d-0.15
150  plot 320+d*cos(i),200+d*sin(i)
160  if xpos=320 and ypos=200 then goto 180
170  next i
180  ink 2,24
190  print"Bitte eine Taste drücken......"
200  taste$=inkey$:if taste$="" then goto 200
210  run
```

Programmbeschreibung:

Zeile 10:	Betriebsmodus 1 anwählen
Zeile 20:	Eingabe der Schrittweite
Zeile 30:	Bildschirm löschen
Zeile 40:	Farbspeicher 2 mit der Farbe Rot füllen
Zeile 50:	Grafikcursor auf die Position 320,200 (Bildschirmmitte)
Zeile 60:	Durchmesser auf 190 setzen
Zeile 70:	Text positioniert ausdrucken
Zeile 80:	"
Zeile 90:	umschalten auf Winkelgradmaß
Zeile 100:	Schleife öffnen
Zeile 110:	Kommentarzeile
Zeile 120:	Wert für die X-Position ausdrucken
Zeile 130:	Wert für die Y-Position ausdrucken
Zeile 140:	Durchmesser um 0.15 verringern
Zeile 150:	Punkt setzen
Zeile 160:	Abfrage, ob die Bildschirmmitte schon erreicht wurde
Zeile 170:	Schleife schließen
Zeile 180:	Schriftfarbe auf Gelb setzen
Zeile 190:	Information auf dem Bildschirm ausgeben
Zeile 200:	Abfrage, ob irgendeine Taste gedrückt wurde
Zeile 210:	Programmstart

Weitere Hinweise:

Experimentieren Sie mit der Schrittweite, dem Durchmesser und der Farbe. Dabei werden Sie jedesmal neue Formen und Farben erzeugen.

Beispiel 5: Karomuster

Aufgabe:

Es soll ein Karomuster, beginnend in der linken unteren Ecke, geplottet werden.

Verarbeitung:

Innerhalb von 2 Schleifen werden die entsprechenden Grafikcursorpositionen angesprochen und auf diesen jeweils ein Bildschirmpunkt gesetzt.

Ausgabe:

Karomuster

Das Programm:

```
10    mode 1
20    input"Schrittweite ?  (max. 50) ",schritt
30    cls
40    move 0,0
50    for i=0 to 200 step schritt
60    for x=0 to 200
70    plot x,i
80    plot i,x
90    next x
100   next i
```

Programmbeschreibung:

Zeile 10: Standardmodus wählen
Zeile 20: Eingabe der Schrittweite
Zeile 30: Bildschirm löschen
Zeile 40: Grafikcursor in die Ausgangsposition bringen
Zeile 50: Schleife öffnen
Zeile 60: innere Schleife öffnen
Zeile 70: an die aktuelle Position einen Einzelpunkt setzen
Zeile 80: an die aktuelle Position einen Einzelpunkt setzen
Zeile 90: innere Schleife schließen
Zeile 100: äußere Schleife schließen

Beispiel 6: Liniengrafik

Aufgabe:

Bildschirmmuster mit PLOT erzeugen

Verarbeitung:

Mit Hilfe von 2 verschachtelten Schleifen soll auf dem Bildschirm ein symmetrisches Muster erzeugt werden.

Ausgabe:

Liniengrafik

Das Programm:

```
10    mode 1
20    cls
30    move 0,0
40    for i=0 to 390
50    for v=0 to 620 step 10
60    plot v,i
70    next v
80    next i
100   for i=0 to 620
110   for h=0 to 390 step 10
120   plot i,h
130   next h
140   next i
150   for farbe=0 to 31
160   ink 2,farbe
170   for pause= 1 to 500
180   next pause
190   next farbe
200   locate 10,15:?"Diese Grafik wurde mit "
210   locate 10,16:?"                       "
220   locate 10,17:?" PLOT erzeugt.        "
230   ink 1,6,24
240   pen 1
250   for pause=1 to 4000
260   next pause
270   run
```

Programmbeschreibung:

Zeile 10: Bildschirmmodus wählen
Zeile 20: Bildschirm löschen
Zeile 30: Grafikcursor in die Ausgangsposition setzen
Zeile 40: Schleife öffnen für 390 vertikale Positionen
Zeile 50: Schleife öffnen für 62 horizontale Positionen
Zeile 60: Einzelpunkt an die berechneten Punkte setzen
Zeile 70: Schleife schließen
Zeile 80: äußere Schleife schließen
Zeile 100: Schleife öffnen
Zeile 110: Schleife öffnen
Zeile 120: Einzelpunkte plotten
Zeile 130: innere Schleife schließen
Zeile 140: äußere Schleife schließen
Zeile 150: Farbschleife öffnen für verschiedene Farben
Zeile 160: in Farbspeicher 2 die Farbe der Schleife setzen
Zeile 170: Warteschleife öffnen
Zeile 180: Warteschleife schließen
Zeile 190: Farbschleife wieder schließen
Zeile 200: positionierte Ausgabe eines Textes

Zeile 210: Leertext ausgeben
Zeile 220: positionierten Text ausgeben
Zeile 230: in Farbspeicher 1 die Farben Rot und Gelb setzen
Zeile 240: Farbregister 1 aufrufen
Zeile 250: Warteschleife öffnen
Zeile 260: Warteschleife wieder schließen
Zeile 270: neuer Programmstart mit RUN

Beispiel 7: Kreis auf der Bildschirmmitte

Aufgabe:

Zeichnen eines Kreises in der Bildschirmmitte (320,200).

Verarbeitung:

Der Ausdruck erscheint auf dem Bildschirm nach der Formel:

```
PLOT X + Radius * cos(i),y + Radius * sin(i)
```

Wenn Sie den Kreis in einer anderen Farbe haben möchten, setzen Sie am Ende der Formel noch die Nummer des gewünschten Farbspeichers.

Ausgabe:

Kreis genau in Bildschirmmitte

Das Programm:

```
10    mode 1
20    cls
30    x=320
40    y=200
50    radius=75
60    for i=1 to 360
70    plot x+radius*cos(i),y+radius*sin(i)
80    next i
```

Programmbeschreibung:

Zeile 10: Standardmodus 1 wählen
Zeile 20: Bildschirm löschen
Zeile 30: Ausgangsposition festlegen
Zeile 40: "
Zeile 50: Radius bestimmen
Zeile 60: Schleife öffnen zum Drucken von 360 Einzelpunkten
Zeile 70: Einzelpunkt berechnen und plotten

Zeile 80: Schleife wieder schließen

Änderungsvorschläge:

Ändern Sie die Position des Kreises, seinen Durchmesser und die Schrittweite der Schleife, die jetzt nur mit 1 arbeitet.
Bestimmen Sie die Farbe des Kreises.

Beispiel 8: Mehrere Kreise

Aufgabe:

Zeichnen von mehreren Kreisen, die ineinander verschlungen sind. Die Kreise sollen alle die gleiche Höhe auf dem Bildschirm haben. Nachdem die Kreise gezeichnet worden sind, sollen alle möglichen Farben einmal durchlaufen werden.

Verarbeitung:

Zeichnen eines Kreises nach der schon bekannten Formel, wobei sich jedesmal bei einem neuen Kreis die X-Position ändert.

Ausgabe:

Mehrere Kreise auf Bildschirmmitte

Das Programm:

```
10    mode 2
20    input"Schrittweite  : ",s
30    cls
40    n=n+1
50    y=200
60    radius=50
70    if n=1 then x=300
80    if n=2 then x=225
90    if n=3 then x=150
100   if n=4 then x=375
110   if n=5 then x=450
120   if n=6 then goto 180
130   deg
140   for i=1 to 360 step s
150   plot x+radius*cos(i),y+radius*sin(i)
160   next i
170   goto 40
180   for i=0 to 31
190   ink 1,i
200   pen 1
210   locate 1,1
220   print"Verwendete Farbnummer : ";i
230   for pause=1 to 500
240   next pause
```

```
250   next i
```

Programmbeschreibung:

Zeile 10: Bildschirmmodus 2 wählen
Zeile 20: Eingabe der Schrittweite
Zeile 30: Bildschirm löschen
Zeile 40: Zähler um 1 erhöhen
Zeile 50: Wert für Y-Achse bestimmen
Zeile 60: Radius bestimmen
Zeile 70: Abfrage, um den X-Wert zubestimmen
Zeile 80: "
Zeile 90: "
Zeile 100: "
Zeile 110: "
Zeile 120: wenn Zähler=6, dann die Farbe wechseln
Zeile 130: Umschalten auf anderes Maßsystem
Zeile 140: Schleife öffnen
Zeile 150: Einzelpunkt berechnen und plotten
Zeile 160: Schleife wieder schließen
Zeile 170: zurück in Zeile 40 gehen
Zeile 180: Schleife für Farbe öffnen
Zeile 190: in Farbspeicher 1 Farbe i setzen
Zeile 200: Schriftfarbe aus Speicher 1 holen
Zeile 210: Textcursor positionieren
Zeile 220: Angabe der verwendeten Farbnummer
Zeile 230: Warteschleife öffnen
Zeile 240: Warteschleife schließen
Zeile 250: Schleife für die Farbe wieder schließen

13.4 Der Grafikbefehl DRAW

Mit diesem Befehl steht Ihnen ein sehr leistungsfähiger Zeichenbefehl zur Verfügung, mit dem Sie auf einfache Weise Linien auf den Bildschirm gezeichnen können. Dieser neue Grafikbefehl hat folgendes Format:

```
DRAW X-Koordinate, Y-Koordinate,
Farbspeichernummer
```

Die ersten beiden Angaben der Koordinaten sind dabei zwingend notwendig, damit der Rechner weiß, wo der Anfangs- und der Endpunkt liegen. Die Angabe des Farbspeichers kann wahlweise erfolgen.

Setzen Sie nun zuerst einmal den CPC in den Einschaltzustand zurück. Wie bereits bekannt, geschieht dies mit gleichzeitigem Drücken der Tasten <SHIFT>, <CTRL> und <ESC>. Geben Sie nun ein:

```
DRAW 0,100
```

Nun erscheint eine Linie am linken unteren Rand des Bildschirms, weil in der linken unteren Ecke der Nullpunkt des Grafikcursors liegt. Mit dem Befehl MOVE kann man diesen Ausgangspunkt aber beliebig wählen.

Da sich nach dem Zeichnen der Linie der Grafikcursor nicht mehr auf seinem Nullpunkt befindet, setzen wir ihn mit MOVE 0,0 wieder zurück in seine Ausgangsposition.

Als nächstes soll eine Linie auf dem unteren Bildschirmrand gezeichnet werden. Dazu geben Sie den Befehl DRAW 200,0 ein, worauf der Computer die Linie ausgibt.

Wenn Sie nun wieder eine Linie nach oben zeichnen möchten, können Sie den DRAW-Befehl erneut einsetzen. Angenommen, Sie möchten ein Rechteck generieren, dann lautet der Befehl dazu:

```
DRAW 200,100.
```

Sie haben jetzt schon drei Linien auf dem Bildschirm. Die vierte und damit letzte Linie entsteht mit DRAW 0,100.

Ein kleines Programm dafür könnte z.B. so aussehen:

```
10    mode 1
20    cls
30    move 0,0
40    draw 0,100
50    move 0,0
60    draw 200,0
70    draw 200,100
80    draw 0,100
90    list
```

Damit Sie die Grafik und das Programm zusammen sehen können, steht in Zeile 90 der Befehl LIST.

Bevor ich Ihnen als nächstes einige Beispiele zeige, probieren Sie selbst am besten einmal aus, welche neuen Möglichkeiten der DRAW-Befehl für Sie bringt. Beachten Sie bei Ihren Versuchen, daß der Grafikcursor nach dem Zeichnen einer Linie immer unsichtbar am Endpunkt dieser Linie steht. Der Endpunkt der alten Linie ist also der Anfangspunkt der neuen Linie. Wenn Sie diese Tatsache berücksichtigen, werden Sie weniger Schwierigkeiten beim Experimentieren haben.

Beispiel 1: Einfacher Rahmen um den Bildschirm

Aufgabe:

Einfachen Rahmen um den Bildschirm zeichnen

Verarbeitung:

Mit DRAW alle 4 Linien zeichnen

Ausgabe:

Feiner Rahmen um den Bildschirmrand

Besonderes:

Sehr schnell

Das Programm:

```
10    mode 1
20    move 0,0
30    cls
40    draw 639,0
50    draw 639,399
60    draw 0,399
70    draw 0,0
80    a$=inkey$:if a$="" then 80
90    locate 10,10
100   print"Ende"
```

Programmbeschreibung:

Zeile 10: Betriebsmodus wählen
Zeile 20: Grafikcursor in Ausgangsposition bringen
Zeile 30: Bildschirm komplett löschen
Zeile 40: Erste Linie zeichnen
Zeile 50: Zweite Linie zeichnen
Zeile 60: Dritte Linie zeichnen
Zeile 70: Vierte Linie zeichnen
Zeile 80: Tastaturabfrage, ob eine Taste gedrückt wurde
Zeile 90: Textcursor auf angegebene Position bringen
Zeile 100: Text ausdrucken

Änderungsmöglichkeiten:

Alle Linien farbig zeichnen (Zeile 40-70 ändern).

Beispiel 2: Farbgrafik-Muster

Aufgabe:

Das folgende Muster soll auf dem Bildschirm erscheinen.

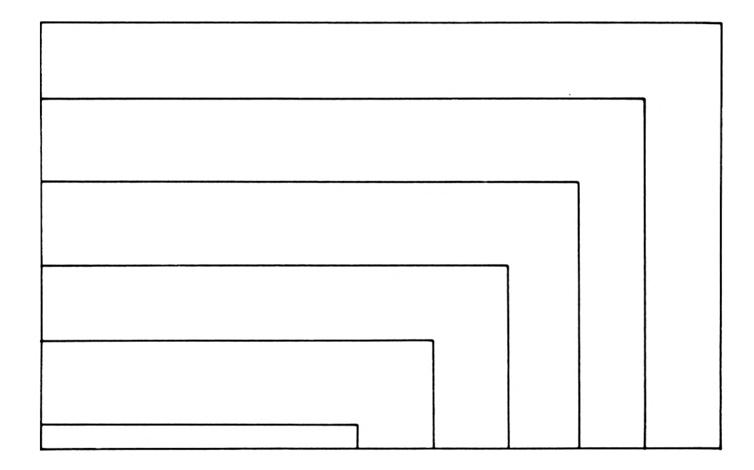


Bild 13.1

Verarbeitung:

Innerhalb einer Schleife werden die Ausgangspositionen immer um einen bestimmten Faktor verringert. Damit wird die dargestellte Grafik immer kleiner.

Ausgabe:

Die oben skizzierte Grafik in zwei verschiedenen Farben auf dem Grafikbildschirm.

Das Programm:

```
10    mode 1
20    cls
30    x=639
40    y=399
50    x=x-10
60    y=y-10
```

```
70    if x<0 then locate 5,10:print"Ende mit 2 mal ESC...": for p=1 to 2000:
      next p:run
80    move 0,0
90    draw x,0
100   draw x,y
110   draw 0,y
120   draw 0,0:goto 50
```

Programmbeschreibung:

Zeile 10: Betriebsmodus wählen
Zeile 20: Bildschirm löschen
Zeile 30: Ausgangskoordinaten setzen
Zeile 40: "
Zeile 50: Koordinate um einen Faktor verringern
Zeile 60: "
Zeile 70: Wenn der Wert für die X-Koordinate unter 0 liegt, dann wird der Textcursor mit **LOCATE** positioniert und anschließend der Text ausgedruckt. Jetzt durchläuft das Programm eine Warteschleife. Dabei hat man Zeit, die <ESC>-Taste zu drücken, um das Programm abzubrechen. Geschieht nichts dergleichen, startet sich das Programm wieder von selbst und beginnt erneut mit der Grafik.
Zeile 80: Bei jedem Durchlauf dieser Zeile wird der Grafikcursor wieder in seine Ausgangsposition gebracht, die sich bekanntlich in der Ecke unten links befindet.
Zeile 90: Zwischen dem berechneten X-Wert und 0 wird eine Linie gezeichnet.
Zeile 100: Als nächstes entsteht eine Verbindung zwischen beiden Punkten X und Y.
Zeile 110: Zwischen 0 und Y wird eine Linie gezeichnet.
Zeile 120: Hier wird eine Linie zum Nullpunkt hin gezogen und das Programm springt zurück in Zeile 50, wo die beiden Koordinaten um einen bestimmten Wert verringert werden.

Änderungsmöglichkeiten:

Als erstes können Sie versuchen, den Abstand der einzelnen Linien zu verändern. Dafür sind die Zeilen 50 und 60 zuständig. Ändern können Sie beispielsweise auch die Ausgangspositionen (Zeilen 30 und 40) und damit die Grafik verkleinern.

Wenn Sie mit der Form der Grafik zufrieden sind, können Sie als nächstes die einzelnen Farben beliebig ändern.

Beispiel 3: Zweifarbige Grafik

Aufgabe:

Es soll eine zweifarbige Grafik entstehen, die auch zu Demonstrationszwecken benutzt werden kann.

Verarbeitung:

Aufbauend auf dem letzten Beispiel, sollen sich hier nun als Blickfang noch einige Linien an einer bestimmten Bildschirmposition um ihre eigene Achse drehen. Auch das Drehen geschieht mit Hilfe des DRAW-Befehls.
Es entsteht später der Effekt einer fast symmetrischen Figur auf dem Bildschirm.

Ausgabe:

Mehrfarbige Grafik wie oben beschrieben

Das Programm:

```
10    mode 1
20    cls
30    border 0
40    ink 0,0
50    move 0,0
60    a=0
70    b=0
80    x=639
90    y=399
100   farbe=int(rnd(1)*27)+1
110   farbe1=int(rnd(1)*27)+1
120   ink 1,farbe
130   ink 2,farbe1
140   n=n+1
150   if n=54 then pen 3:locate 30,22:print" Ende": for i=1 to 5000:next i:run
160   x=x-10
170   y=y-10
180   a=a+10
190   b=b+10
200   move a,b
210   draw x,0,1
220   draw x,y,2
230   draw 0,y,1
240   draw a,b,2
250   goto 140
```

Programmbeschreibung:

Zeile 10: Betriebsmodus wählen
Zeile 20: Bildschirm löschen
Zeile 30: Rahmen in Schwarz einfärben

Zeile 40:	Bildschirmhintergrund in Schwarz färben
Zeile 50:	Grafikcursor in Ausgangsposition bringen
Zeile 60:	Bestimmen der Ausgangsgrößen
Zeile 70:	"
Zeile 80:	"
Zeile 90:	"
Zeile 100:	Die erste Farbe errechnen
Zeile 110:	Die zweite Farbe errechnen
Zeile 120:	Die ermittelte Farbe in Farbspeicher Nr. 1 setzen
Zeile 130:	Die ermittelte Farbe in Farbspeicher Nr. 2 setzen
Zeile 140:	Zähler bei jedem Durchlauf um 1 erhöhen
Zeile 150:	Wenn der Zähler den Wert 54 erreicht hat, dann den Cursor mit **LOCATE** positionieren und das Wort "Ende" ausdrucken lassen. Anschließend eine Warteschleife durchlaufen und dann das Programm mit **RUN** erneut starten.
Zeile 160:	X-Koordinate um Faktor 10 verringern
Zeile 170:	Y-Koordinate um Faktor 10 verringern
Zeile 180:	Wert für A um 10 erhöhen
Zeile 190:	Wert für B um 10 erhöhen. Die beiden Angaben (A und B) sind für die Drehung der Linien verantwortlich.
Zeile 200:	Grafikcursor positionieren
Zeile 210:	Linie zeichnen zwischen den angegebenen Koordinaten mit der Farbe aus Farbspeicher Nr. 1
Zeile 220:	wie Programmzeile 210, hier wird die Zeichenfarbe jedoch aus Farbspeicher Nr. 2 geholt
Zeile 230:	wie Programmzeile 210
Zeile 240:	wie Programmzeile 220
Zeile 250:	Rücksprung in Zeile 140, wo der Zähler um 1 erhöht wird.

Änderungsmöglichkeiten:

Im Prinzip gibt es hier, ebenso wie im letzten Beispielprogramm, viele Möglichkeiten.

Hinweis:

Die Zahl 54 wurde von mir experimentell ermittelt. Diese Zahl habe ich verwendet, weil sich bei 54 beide drehenden Linien treffen.

Beispiel 4: Farbiger Bildschirmrahmen

Aufgabe:

Ein Rahmen soll mit dem Befehl DRAW um den Bildschirm gezeichnet werden. Das Programm soll Schritt für Schritt zeigen, wie die einzelnen Linien entstehen.

Verarbeitung:

Zuerst wird die untere Linie von links nach rechts gezeichnet. Anschließend verzweigt das Programm in ein Unterprogramm und wartet hier auf einen Tastendruck.

Als nächstes entsteht die rechte Linie von unten nach oben. Ist dies geschehen, wird wieder das Unterprogramm aufgerufen.

Danach kommt die obere Linie, die von links nach rechts gezeichnet wird. Wie bei den Vorgängern verzweigt das Programm hier wieder in das Unterprogramm und wartet auf einen Tastendruck.

Wird dann eine Taste betätigt, entsteht die letzte Linie. Sie verbindet die linke obere Ecke mit dem Nullpunkt des Grafikcursors.

Alle Linien sollen farbig sein. Damit man als Anwender auch die Arbeitsweise des Programms verfolgen kann, wird jede Linie noch bezeichnet.

Ausgabe:

Farbiger Rahmen um den Bildschirm

Das Programm:

```
10    mode 1
20    cls
30    move 0,0
40    draw 639,0,1
50    gosub 150
60    draw 639,399,2
70    gosub 150
80    draw 0,399,3
90    gosub 150
100   draw 0,0,5
110   gosub 150
120   locate 2,20
130   print"Programm mit 2 mal ESC beenden....."
140   goto 140
150   rem  Unterprogramm für die Bezeichnung
160   rem  der Linien und Tastaturabfrage,
170   rem  ob eine Taste gedrückt wurde
180   read n$
```

```
190    locate 10,10
200    print"Das war die ";n$;" Linie."
210    for i=1 to 5
220    print chr$(7);
230    for pause=1 to 200:next pause
240    next i
250    a$=inkey$:if a$="" then goto 250
260    return
270    rem  Ende des Unterprogramms
280    rem
290    rem  DATA-Werte für die Linien
300    data erste,zweite,dritte,vierte
```

Programmbeschreibung:

Zeile 10: Betriebsmodus anwählen
Zeile 20: Bildschirm löschen
Zeile 30: Grafikcursor in seine Ausgangsposition bringen
Zeile 40: Erste Linie zeichnen (unten)
Zeile 50: Unterprogramm zur Bezeichnung der Linie aufrufen
Zeile 60: Zweite Linie wird gezeichnet (rechts)
Zeile 70: Unterprogramm aufrufen
Zeile 80: Dritte Linie zeichnen (oben)
Zeile 90: Unterprogramm aufrufen
Zeile 100: Vierte Linie zeichnen (links)
Zeile 110: Unterprogramm aufrufen
Zeile 120: Textcursor auf angegebene Position positionieren
Zeile 130: Text ausdrucken : "Programm mit 2mal <ESC> beenden"
Zeile 140: Endlosschleife, die nur mit ESC unterbrochen werden kann
Zeile 150: Kommentarzeile
Zeile 160: "
Zeile 170: "
Zeile 180: Bezeichnung der Linie aus den DATA-Angaben einlesen
Zeile 190: Textcursor auf die angegebene Position setzen
Zeile 200: Bezeichnung der Linie
Zeile 210: Schleife 5mal durchlaufen
Zeile 220: Piepton erzeugen
Zeile 230: Warteschleife durchlaufen
Zeile 240: Schleife schließen
Zeile 250: Tastaturabfrage; wenn keine Taste gedrückt wird, dann weiter abfragen. Wird eine Taste gedrückt, geht es mit der nächsten Zeile weiter.
Zeile 260: Rücksprung ins Hauptprogramm
Zeile 270: Kommentarzeile
Zeile 280: "

Zeile 290: "
Zeile 300: **DATA**-Werte für die Bezeichnung der Linien

Beispiel 5: Stern

Aufgabe:

Es soll ein Stern in der Bildschirmmitte gezeichnet werden.

Verarbeitung:

In die Bildschirmmitte (320,200) wird der Grafikcursor plaziert. Anschließend wird mit Hilfe eines immer kleiner werdenden Radius und einer Schleife der Stern erstellt. Der Rechner muß dazu mit RAD in den Bogenmaßradius umgeschaltet werden.

Nachdem der Stern gezeichnet worden ist, soll er noch in verschiedenen Farben auf dem Bildschirm blinken.

Ausgabe:

Fertig gezeichneter Stern

Das Programm:

```
10    mode 1
20    cls
30    ende=int(rnd(1)*1600)+400
40    border 0
50    z=int(rnd(1)*27)+1
60    ink 1,z
70    ink 0,0
80    ink 2,6
90    move 320,200
100   d=200
110   for i=1 to ende step 4
120   d=d-0.5
130   rad
140   if i<200 then f=1 else f=6
150   draw 320+d*cos(i),200+d*sin(i),f
160   next i
170   for i=1 to 30 step 2
180   ink 1,i
190   for i2=0 to 31
200   ink 2,i2
210   for pause=1 to 100
220   next pause
230   next i2
240   next i
250   run
```

Programmbeschreibung:

Zeile 10:	Bildschirmmodus wählen
Zeile 20:	Bildschirm löschen
Zeile 30:	Endwert für eine Schleife bilden. Dieser Endwert bestimmt später, wie groß der Stern wird.
Zeile 40:	Bildschirmrahmen in Schwarz umfärben
Zeile 50:	Farbe für den Stern wählen. Die hinzuaddierte 1 am Ende der Zeile verhindert, daß Schwarz als Zeichenfarbe gewählt wird. Dies ist nötig, da in der nächsten Zeile der Bildschirmhintergrund die Farbe Schwarz erhält.
Zeile 60:	Die Farbe Schwarz in den Farbspeicher 0 bringen. Der Bildschirmhintergund wird damit schwarz.
Zeile 70:	Die errechnete Zeichenfarbe aus Zeile 50 wird in den Farbspeicher 1 gebracht. Damit liegt die Zeichenfarbe zum Zeichnen des Sterns abrufbar in Farbspeicher 1 bereit.
Zeile 80:	Die Farbe Nr. 6, die der Farbe Hellrot entspricht, wird in den Farbspeicher 2 gebracht.
Zeile 90:	Der Grafikcursor wird in der Mitte positioniert.
Zeile 100:	Die Größe des Sterns wird mit dieser Zeile festgelegt.
Zeile 110:	Schleife zum Zeichnen öffnen mit einer bestimmten Schrittweite
Zeile 120:	Die Größe des Durchmessers wird bei jedem Durchlauf der Schleife um einen bestimmten Wert verringert. Der Stern beginnt also als großes Objekt und wird bei jedem Schleifendurchlauf immer kleiner.
Zeile 130:	Ins Bogenmaß umschalten
Zeile 140:	Wenn das Bogenmaß die Schleife 200 mal durchlaufen hat, dann wird mit einer anderen Farbe weiter gezeichnet. Dies bedeutet, bis zum zweihundertsten Durchlauf wird der Stern mit der Farbe gezeichnet, die in Farbspeicher 1 steht. Anschließend wird mit der Farbe Hellrot weitergezeichnet, da sich diese Farbe besonders gut für Effekte eignet.
Zeile 150:	Stern zeichnen mit der in Zeile 140 ermittelten Farbe, bzw. der Farbe, die in Farbspeicher Nr. F steht.
Zeile 160:	Schleife schließen
Zeile 170:	Schleife für verschiedene Farben öffnen
Zeile 180:	Farbspeicher Nr. 1 die Farbnummer zuweisen, die durch die Schleife ermittelt wurde.
Zeile 190:	Zweite Schleife für Farbe öffnen
Zeile 200:	Aktuelle Farbe in den Farbspeicher Nr. 2 bringen

Zeile 210: Warteschleife öffnen
Zeile 220: Warteschleife schließen
Zeile 230: Zweite Schleife für Farbe wieder schließen
Zeile 240: Erste Schleife für Farbe wieder schließen
Zeile 250: Programm erneut starten

Änderungsmöglichkeiten:

Ändern Sie den Befehl DRAW in Zeile 150 in PLOT um. Dazu ist auch noch eine Änderung der Zeile 110 interessant. Sie müßte dann so aussehen:

```
110 for i=1 to ende step 1.5
```

Durch wenige Änderungen ergibt sich bereits ein völlig neues Bild.

13.5 Relatives Zeichnen

Erschrecken Sie bitte nicht vor dem Begriff "relativ". Dahinter versteckt sich eine ganz einfache Zeichenmethode, die wir uns einmal näher ansehen wollen. Das nachfolgende Beispiel wird uns dabei helfen, die Arbeitsweise beim relativen Zeichnen zu verstehen.

Bitte sehen Sie sich folgende Aufgabenstellung an:

Es sollen sieben Linien auf dem Bildschirm dargestellt werden. Dabei soll beim Zeichnen der Grafikbefehl PLOT herangezogen werden.

Ein solches Programm haben wir bereits einmal erarbeitet und zwar als der Befehl PLOT vorgestellt wurde.

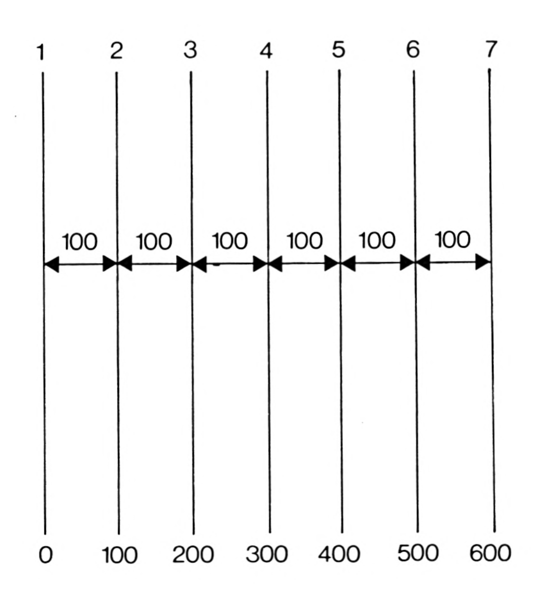


Bild 13.2

Hier nochmals zur Erinnerung der wichtigste Programmteil dieses Beispiels, nämlich derjenige Teil, der für das Zeichen der sieben Linien verantwortlich ist.

```
50    for i=0 to 399
60    plot 0,i
70    plot 100,i
80    plot 200,i
90    plot 300,i
100   plot 400,i
110   plot 500,i
120   plot 600,i
130   next i
```

Beachten Sie die Zeilen 60 bis 120. Hier bezieht sich der Rechner immer auf den Nullpunkt in der linken Ecke des Bildschirms.

Um beispielsweise die siebente Linie zu zeichnen, geht der Rechner, vom Nullpunkt aus gerechnet, 600 Positionen weit nach rechts und beginnt an dieser Stelle mit dem Zeichnen der Linie. Dieses Verfahren könnte man auch als absolutes Zeichnen bezeichnen. Der CPC bezieht sich dabei immer auf einen festen Ausgangspunkt.

Unter "relativem Zeichnen" versteht man nun folgendes:

a. Der Bezugspunkt ist jetzt flexibel.

b. Der neue Bezugspunkt ist identisch mit den alten Koordinaten der eben gezeichneten Grafik.

Im Klartext bedeutet das, daß sich der Bezugspunkt nicht mehr links unten befindet, sondern als neuer Nullpunkt die Koordinaten der letzten gezeichneten Linie angenommen wird.

Ich werde Ihnen im folgenden den Unterschied zwischen absolutem und relativem Zeichnen am praktischen Beispiel vorführen. Wie Sie aus der Abbildung entnehmen konnten, haben alle Linien den gleichen Abstand zueinander (100). Wie absolut gezeichnet wird, haben wir eben bereits gesehen. Der Rechner bezieht sich bei allen Koordinateneingaben immer auf den Nullpunkt in der linken unteren Ecke.

Und nun zum relativen Zeichnen. Nachdem eine Linie gezeichnet wurde, nimmt der CPC diese Position der eben erstellten Linie als neuen Nullpunkt an. Dann wird mit dem Befehl **PLOTR** dem Rechner gesagt, er soll von der aktuellen Position aus 100 Positionen weiter nach rechts gehen und dort mit der nächsten Linie fortfahren. Nachfolgend das Programmlisting:

```
10    mode 1
20    cls
30    print"Relatives Zeichnen mit dem CPC 464"
40    move 0,0
50    for i=0 to 399
60    plot 0,i
70    plotr 100,i
80    plotr 100,i
90    plotr 100,i
100   plotr 100,i
110   plotr 100,i
120   plotr 100,i
130   next i
```

Wie Sie sehen, ist das Arbeiten mit relativen Koordinatenangaben nicht so schwierig, wie es am Anfang schien.

In dem eben gezeigten Beispiel wurde der Befehl **PLOTR** verwendet, um eine Linie zu erzeugen. Außer diesem Befehl stehen auch noch die unten aufgeführten anderen Grafikbefehle für relatives Zeichnen zur Verfügung. Sie erkennen sie an einem angehängten "R".

DRAWR	zeichnet eine Linie, von der letzten Position des Grafikcursors aus gerechnet
MOVER	setzt den Grafikcursor von der letzten Position des Cursors aus gerechnet weiter
TESTR	gibt die verwendete Farbspeichernummer einer farbigen Grafik vom relativen Bezugspunkt aus gerechnet an

Neben den Befehlen, wie PLOTR 100,I oder DRAWR 100,250, gibt es Befehle mit solchen oder ähnlichen Angaben:

PLOTR 100,I	vom relativen Bezugspunkt aus gesehen, 100 Positionen nach links gehen und einen Punkt setzen
MOVER 200,100	setzt z.B. den Grafikcursor um 200 Positionen, von seiner aktuellen Position aus gesehen, zurück

Also: Bei negativen Angaben geht der Grafikcursor immer zurück.

Sicherlich werden Sie eine ganze Weile brauchen, bis Sie genügend Erfahrung haben, um die Befehle für das relative Zeichnen voll einsetzen zu können. Haben Sie etwas Geduld, und lassen Sie Ihrer Phantasie freien Lauf; Sie werden dann erstaunt sein, welche interessanten Grafiken Sie gestalten können.

Beispiel: Bildschirmrahmen

Aufgabe:

Rahmen um den Bildschirm zeichnen

Verarbeitung:

Mit Hilfe einer Schleife sowie den Befehlen PLOT und PLOTR die obere und untere Linie sowie rechte und linke Linie gleichzeitig zeichnen.

Ausgabe:

Rahmen um den Bildschirm

Das Programm:

```
10    mode 1
20    cls
30    move 0,0
40    rem ----------------------------------------------------
50    rem  obere und untere Linie zeichnen
60    for i=0 to 639
70    plot i,0,2
80    plotr 0,399,2
90    next i
```

```
100   rem  Linie oben und unten gezeichnet
110   rem -----------------------------------------------------
120   rem
130   rem
140   rem
150   rem
160   rem
170   rem  rechte und linke Linie zeichnen
180   move 0,0
190   for i=0 to 399
200   plot 0,i,2
210   plotr 639,0,2
220   next i
230   rem  Linie rechts und links fertig
240   rem -----------------------------------------------------
250   locate 15,10
260   print"Rahmen fertig !"
270   for i=1 to 10
280   print chr$(7)
290   for pause=1 to 300
300   next pause
310   next i
320   locate 2,15
330   print"Programm mit 2 mal ESC abbrechen...."
340   goto 340
```

Programmbeschreibung:

Das eben vorgestellte Programm dokumentiert sich zum größten Teil selbst.

Wichtig für das relative Zeichnen sind die Zeilennummern 80 und 210.

Der Bereich ab Zeilennummer 250 dient lediglich als Hinweis, daß der Rahmen gezeichnet worden ist.

Die Zeile 280 erzeugt einen kurzen Piepton und entspricht dem Kontrollzeichen BELL. Dazu können Sie Kapitel 9 des Originalhandbuchs aufschlagen, wo Sie auf Seite 2 weitere Informationen zum Thema Kontrollzeichen finden.

13.6 Zeichen und Texte in Grafiken

Vielleicht haben Sie auch schon einmal versucht, innerhalb einer Grafik an einer ganz bestimmten Stelle ein Zeichen oder einen kleinen Text zu setzen?

Sie haben dafür folgende Möglichkeiten:

a) im Modus 0 stehen Ihnen 500 verschiedene Positionen zur Verfügung (20 Zeichen pro Zeile mal 25 mögliche Zeilen)

b) im Modus 1 stehen Ihnen 1000 verschiedene Positionen zur Verfügung (40 Zeichen pro Zeile mal 25 mögliche Zeilen)

c) im Modus 2 stehen Ihnen maximal 2000 verschiedene Positionen zur Verfügung (80 Zeichen pro Zeile mal 25 mögliche Zeilen)

Vermutlich werden Sie denken: "2000 verschiedene Schreibpositionen? Das müßte normalerweise doch wirklich ausreichen!" Genau diesen Gedanken hatte ich am Anfang auch.

Doch nach und nach kam ich zu der Erkenntnis, daß es doch ganz gut wäre, wenn man seine Schreibpositionen frei wählen könnte.

Wenn ich hier von frei spreche, meine ich eine Position auf dem Grafikbildschirm, die genauso angesteuert werden kann, wie ein einzelner Punkt mit dem Befehl **PLOT**. Ihnen ist sicherlich bekannt, daß es einen Text- und einen Grafikcursor gibt. Auf dem normalen Bildschirm können Sie nur mit dem Befehl **LOCATE** einen bestimmten Punkt ansteuern und ab dieser Stelle einen Text beginnen.

Leider wirkt der Befehl **LOCATE** nicht auf dem Grafikbildschirm; dafür gibt es einen speziellen Befehl. Mit **TAG** teilt man dem Rechner mit, daß man ein Zeichen oder einen Text an einer ganz bestimmten Stelle auf dem Grafikbildschirm ausgeben möchte.

Diese Stelle wird mit Hilfe des Grafikcursors angesteuert (MOVE).

Nachdem der Grafikcursor mit **MOVE** plaziert wurde, kann ein Text geschrieben werden. Ihren Text oder andere Zeichen können Sie jetzt wie gewohnt mit dem Befehl **PRINT** auf den Grafikbildschirm bringen.

Dazu gleich ein praktisches Beispiel. Versetzen Sie zunächst den Rechner in den Einschaltzustand mit <ESC>, <CTRL> und <SHIFT>. Anschließend geben Sie ein:

```
10    mode 1
20    tag
30    move 100,100
40    print"Dieser Text erscheint auf dem Grafikbildschirm";
50    tagoff
60    print"Ende"
```

Das kleine Programm demonstriert Ihnen, daß es möglich ist, mit dem Befehl **PRINT** auf dem Grafikbildschirm beliebige Zeichen und Texte auszugeben.

Und nun zur Beschreibung des Programms:

Zeile 20:	mit TAG dem Rechner mitteilen, daß auf dem Grafikbildschirm mit dem Befehl PRINT ein Text erscheinen soll
Zeile 30:	Grafikcursor mit MOVE 100,100 plazieren
Zeile 40:	beliebigen Text mit PRINT auf den Bildschirm bringen
Zeile 50:	mit TAGOFF dem Rechner mitteilen, daß jetzt wieder auf dem normalen Bildschirm geschrieben werden soll
Zeile 60:	der Text in dieser Zeile erscheint wieder auf dem normalen Bildschirm

Wichtig ist das Semikolon am Ende der PRINT-Anweisung in Zeile 40! Fehlt dieses Semikolon, steht später ein sogenanntes Kontrollzeichen am Ende der Zeichenkette. Versuchen Sie es ruhig einmal und entfernen das Semikolon.

Die Einrichtung, in einer Grafik auch beliebige Zeichen erscheinen lassen zu können, macht den CPC besonders interessant für Grafikanwendungen. Bei anderen Homecomputern dieser Klasse ist dies nicht immer möglich!

An dieser Stelle möchte ich Ihnen einmal schildern, wie umständlich es teilweise beim Commodore 64 ist, die durchaus gute Grafik voll zu nutzen. Wenn Sie bei diesem Computer eine Grafik beschriften wollen, müssen Sie die folgende Arbeitsweise anwenden:

a) Sie erstellen sich mit POKE-Befehlen Ihre Grafik, denn spezielle Grafikbefehle fehlen in der Grundausstattung völlig.

b) Da sich der Standardzeichensatz nicht auf dem Grafikbildschirm darstellen läßt, muß ein spezieller Zeichensatz zur Verwendung in Grafiken erstellt werden.

c) Der neue Zeichensatz verbraucht natürlich Speicherplatz. Auch diese Tatsache darf man nicht außer acht lassen!

Im Gegensatz zum Commodore 64 sind die Grafikeigenschaften beim Schneider doch recht zufriedenstellend. Wünschenswert wären vielleicht noch weitere Befehle, die die guten Grafikeigenschaften unterstützen, wie etwa ein Befehl, mit dessen Hilfe man Kreise oder Rechtecke zeichnen kann.

Mit Sicherheit werden jedoch in der nächsten Zeit viele sogenannte Befehlserweiterungen, in denen diese und viele andere Befehle enthalten sind, auf den Markt kommen. Hier zeichnet sich eine ähnlich Entwicklung ab wie beim Commodore 64.

14 Arbeiten mit dem Standardzeichensatz

In jedem Computersystem dieser Leistungsklasse gibt es einen Zeichensatz, der aus 256 verschiedenen Zeichen und Steueranweisungen besteht. Diesen Zeichensatz nennt man ASCII-Code.

Der erste Bereich des ASCII-Codes von 0 bis 31 besteht aus sogenannten Steuerfunktionen, die beispielsweise dazu dienen, den Cursor um eine Position zurückzusetzen, den Cursor in die linke obere Ecke zu setzen oder Inverse Darstellung einzuschalten.

In diesem Kapitel interessieren uns aber weniger die Funktionen als vielmehr der Zeichensatz und welche Zeichen sich damit darstellen lassen. Beginnen wir nun mit der ersten Gruppe von Zeichen, im Bereich von 32 bis 47. Dieser Bereich enthält Sonderzeichen und mathematische Symbole. Ab Code-Nr. 48 folgen dann Ziffern, Satzzeichen und Vergleichsoperatoren.

Das Alphabet beginnt bei Code-Nr. 65 mit einem großen A und hört bei Code-Nr. 90 mit dem Großbuchstaben Z auf. Daran schließt sich ein Bereich an, der sich für spätere Zeichensatzerweiterungen geradezu anbietet. Hier liegen nämlich, angefangen bei Code-Nr. 91 bis zur Code-Nr. 96, einige Sonderzeichen, auf die man gut verzichten kann. Aus diesem Grund besteht im Bereich 91 bis einschließlich 96 die Möglichkeit, den deutschen Zeichensatz einzubauen.

Der Schneider enthält als Computer mit ASCII-Zeichensatz, also einer USA-Norm, keine deutschen Zeichen, das hat zur Folge, daß die Umlaute und das ß fehlen. Wie man selbst eigene Zeichen erstellen kann, wird im nächsten Kapitel erläutert.

Doch nun zurück zu unserem Standardzeichensatz. Wir haben soeben den Bereich von 91 bis 96 kennengelernt. Ab Code-Nr. 97 beginnt dann das kleine Alphabet, das bis Code-Nr. 122 reicht. Ab Code-Nr. 123 folgt der interessanteste Bereich. Neben sehr vielen Grafikzeichen findet man dort eine Reihe von Sonderzeichen, die sich gut in eigenen Programmentwicklungen unterbringen lassen.

Mich persönlich stört dabei allerdings die Vielzahl der griechischen Zeichen. Stattdessen wäre der deutsche Zeichensatz für viele Anwender sicherlich interessanter gewesen. Aber wie Sie bereits wissen, können Sie ja ohne großen Aufwand neue Zeichen zu erstellen. Wollen Sie sich die Zeichen einmal ansehen, geben Sie dazu das folgende Programm ein:

```
10    MODE 1
20    CLS
30    FOR I=32 TO 255
40    PRINT"ASCII-CODE : ";I;"   ZEICHEN:   ";CHR$(I)
50    PRINT
60    A$=INKEY$: IF A$="" THEN 60
70    NEXT I
```

Nachfolgend werden einige Programme aufgelistet, in denen in erster Linie die Grafikzeichen vorkommen, mit denen Sie Grafiken auf dem Bildschirm erstellen können. Nichts ist zeitraubender, als eine Grafik Zeichen für Zeichen zusammenzusetzen!

Im Anhang Ihrer Bedienungsanleitung für den CPC 464 finden Sie drei Bildschirmarbeitsblätter. Sie können sich also aussuchen, ob Sie das Blatt **MODE 0**, **MODE 1** oder **MODE 2** verwenden möchten, je nachdem, welcher Bildschirmmodus gerade am besten geeignet ist. Bevor Sie mit Ihrer Programmierarbeit beginnen, sollten Sie die Grafik zunächst auf Papier vorzeichnen, da sich so die Koordinaten wesentlich schneller ermitteln lassen, als wenn Sie lange am Bildschirm herumprobieren.

14.1 Menüaufbau

Dieses Programm demonstriert Ihnen, wie man mit einfachen Grafikzeichen ein Menübild programmieren kann. Sie können das Beispiel in Ihre eigenen Programme übernehmen, da ein übersichtlicher Programmaufbau immer sowohl für Sie als auch für einen fremden Anwender von Vorteil ist.

```
100   MODE 1
110   CLS
120   BORDER 0
130   INK 0,0
140   PEN 2
150   MI$=STRING$(5,154)+CHR$(159)+STRING$(32,154)
160   O$=CHR$(150)+STRING$(5,154)+CHR$(158)+STRING$(32,154)+CHR$(156)
170   U$=CHR$(150)+STRING$(5,154)+CHR$(155)+STRING$(32,154)+CHR$(153)
180   M$=CHR$(149)
190   PRINT O$
200   FOR P=2 TO 22
```

```
210   LOCATE 1,P
220   PRINT M$
230   LOCATE 7,P
240   PRINT M$
250   LOCATE 40,P
260   PRINT ,$
270   NEXT
280   LOCATE 1,22
290   PRINT U$
300   FOR I= 4 TO 20 STEP 2
310   LOCATE 1,I
320   PRINT CHR$(151);
330   PRINT MI$;
340   LOCATE 40,I
350   PRINT CHR$(157);
360   NEXT
370   LOCATE 1,24
380   REM
390   REM  - - - - - Menue bezeichnen - - - - -
400   REM
410   DATA 1985
420   DATA H A U P T - M E N U E
430   DATA 1, Informationen zum Programm
440   DATA 2, Daten von Kassette/Diskette laden
450   DATA 3, Daten eingeben
460   DATA 4, Daten auf Kassette/Diskette speichern
470   DATA 5, Daten sortieren
480   DATA 6, Daten drucken und listen
490   DATA 7, Daten auswerten
500   DATA 8, Daten ändern
510   DATA 9, Programm beenden
520   REM
530   REM
540   REM
550   REM  - - - Daten lesen und in die - - - -
560   REM
570   REM  - - - Bildschirmmaske bringen - - -
580   REM
590   REM
600   READ JAHR$
610   LOCATE 2,3
620   PRINT CHR$(164); JAHR$
630   PEN 3
640   READ BEZEICHNUNG$
650   LOCATE 10,3
660   PRINT BEZEICHNUNG$
670   PEN 1
680   FOR I=5 TO 21 STEP 2
690   READ ZAHL
700   LOCATE 3, I
710   PRINT ZAHL
720   READ BEZEICHNUNG
730   LOCATE 10,I
740   PRINT BEZEICHNUNG$
750   NEXT
760   REM  - - - - - jetzt kommt die Eingabe - - -
770   PRINT"Bitte geben Sie die gewünschte Zahl ein......."
```

```
780   A$=INKEY$: IF A$="" THEN 780
790   IF A$="1" THEN GOTO 1000
800   IF A$="2" THEN GOTO 2000
810   IF A$="3" THEN GOTO 3000
820   IF A$="4" THEN GOTO 4000
830   IF A$="5" THEN GOTO 5000
840   IF A$="6" THEN GOTO 6000
850   IF A$="7" THEN GOTO 7000
860   IF A$="8" THEN GOTO 8000
870   IF A$="9" THEN CLS:END
880   REM
890   REM  - - - wenn Eingabe nicht den Abfragen entspricht,
900   REM  - - - dann hier weiter machen - - -
910   SOUND 1,800,90,7
920   SOUND 2,801,90,7
930   GOTO 780
```

Programmbeschreibung:

Zeile 100 bis 140: Bildschirmmodus und Farben einstellen
Zeile 150 bis 180: Zeichenketten aus Grafikzeichen zusammensetzen
O$ = Obere Linie
MI$= Mittlere Linien
U$ = Untere Linie
M$=Grafikzeichen für den zeitlichen Rahmen
Zeile 190: Obere Linie drucken
Zeile 200 bis 270: Alle vertikalen Linien zeichnen
Zeile 280 und 290: Untere Linie zeichnen
Zeile 300 bis 360: Alle mittleren Linien einzeichnen
Zeile 370 bis 400: Neu positionieren und Kommentarzeilen
Zeile 410 bis 510: Bezeichnungen für die Bildschirmmaske bereitstellen
Zeile 600: Lesen der ersten Ausgabe (die Jahreszahl)
Zeile 610: Cursor oben positionieren
Zeile 620: Copyrightzeichen und Jahresangabe ausgeben
Zeile 630: Auf andere Zeichenfarbe umschalten
Zeile 640: Das Wort H A U P T - M E N U E lesen
Zeile 650: Cursor neu positionieren
Zeile 660: Das Wort H A U P T - M E N U E ausgeben
Zeile 670: Die Zeichenfarbe wieder wechseln
Zeile 680 bis 750: Schleife zum Lesen und Drucken der letzten Bezeichnungen auf dem Bildschirm
Vorgehensweise:
a) Zahl lesen'
b) Cursor vorne positionieren
c) Zahl ausdrucken
d) Bezeichnung lesen

e) Cursor weiter nach hinten positionieren
f) Bezeichnung ausdrucken
g) Nächsten Datenbestand lesen und damit ebenso verfahren

Zeile 760: Kommentarzeile
Zeile 770: Ausgaben, was als nächstes getan werden soll
Zeile 780: Tastaturabfrage, ob ein Zeichen eingegeben wurde
Zeile 790 bis 870: Bei Eingabe einer 1, 2, 3, 4, 5, 6, 7, 8 oder 9 die entsprechenden Programmteile mit GOTO anspringen. Bei der Eingabe der Zahl 9 den Bildschirm löschen und das Programm beenden.
Zeile 880 bis 930: Falls das eingegebene Zeichen nicht den gewünschten Ziffern entsprach, dann einen Warnton erzeugen und über den Sprung auf Zeile 780 nach einem neuen Zeichen fragen.

Das Programm ist recht lang, dafür aber auch sehr übersichtlich und änderungsfreundlich. Wenn Sie z.B. andere Bezeichnungen haben möchten, brauchen Sie nur in die Zeilen 410 und 510 Ihre eigenen Angaben eingeben. Falls Sie andere Anfangszeilennummern für Ihre Programmteile benötigen, erhalten Sie diese, indem Sie in den Zeilennummern 790 bis 870 die Sprungadressen entsprechend ändern.

Ist dann Ihr persönliches Programm-Menü fertig, sollten Sie damit beginnen, in einer Zeilennummer mehrere Befehle unterzubringen. Damit können Sie das Programm zwar erheblich verkleinern, was aber auf Kosten der Übersichtlichkeit geht.

Am besten ist es, wenn Sie das Programm in der ausführlichen Form speichern. Wenn Sie dann wieder einmal ein Menü brauchen, holen Sie es sich von der Kassette oder Diskette in den Speicher, verändern es nach Bedarf und komprimieren es. Es ist kaum möglich, noch schneller an einsatzfähige, fast fertige Programme heranzukommen.

14.2 Eingabemaske einer Kundenkartei

Das nun folgende Programm soll Ihnen zeigen, wie man recht einfach eine Eingabemaske zur Datenerfassung programmieren kann. Als Beispiel wurde dabei eine Kundenkartei ausgewählt, weil diese für jeden verständlich ist. Eingabekriterien sind:

a) Kundennummer
b) Kundenname
c) Straße und Hausnummer des Kunden
d) Postleitzahl und Wohnort
e) Telefonnummer
f) Werbekennzeichen
g) Ansprechpartner
h) Umsatz Vorjahr

Zunächst noch ein paar Worte zu den Kriterien Werbekennzeichen, Ansprechpartner und Umsatz Vorjahr. Das Werbekennzeichen wird für spätere Werbeaktionen benötigt. Hat man z.B. gerade ein günstiges Angebot für eine bestimmte Zielgruppe, dann können alle Kunden mit dem Werbekennzeichen XY angesprochen werden. Damit wird gewährleistet, daß spezielle Waren auch nur einem speziellen Kundenkreis angeboten werden. Zusätzlich können die sowieso schon sehr hohen Werbungskosten erheblich dezimiert werden.

Der Ansprechpartner ist für Verhandlungen wichtig, die man mit diesem Kunden immer führt. Der Umsatz des Vorjahres sagt aus, ob es sich um einen guten oder um einen weniger guten Kunden handelt. Bei Lieferengpässen wird dann dem Kunden mit dem höheren Umsatz wahrscheinlich eher der Zuschlag gegeben, als einem Kunden, der nur gelegentlich bestellt.

Es folgt nun das Programm zur Erfassung von Kundenanschriften.

```
100   MODE 1
110   CLS
120   BORDER 0
130   INK 0,0
140   PEN 2
150   I$=CHR$(24)
160   O$=CHR$(150)+STRING$(38,154)+CHR$(156)
170   U$=CHR$(147)+STRING$(38,154)+CHR$(153)
180   M$=CHR$(149)
190   PRINT O$
200   FOR I=2 TO 22
210   LOCATE 1,I
220   PRINT M$
230   LOCATE 40,I
240   PRINT M$;
250   NEXT
260   PRINT U$
270   LOCATE 2,5
280   FOR I=2 TO 39
290   LOCATE I,4
300   PRINT"_"
310   NEXT
320   REM
```

```
330   REM    - - - - Rahmen gezeichnet - - - -
340   REM
350   PEN 3
360   LOCATE 8,3:PRINT"K U N D E N K A R T E I "
370   PEN 1
380   LOCATE 2,6:PRINT"Kundennummer:"
390   LOCATE 2,8:PRINT"Kundenname:"
400   LOCATE 2,10:PRINT"Str. und Nr.:"
410   LOCATE 2,12:PRINT"PLZ und Ort:"
420   LOCATE 2,14:PRINT"Telefonnummer:"
430   LOCATE 2,16:PRINT"Werbekennzeichen:"
440   LOCATE 2,18:PRINT"Umsatz Vorjahr:"
450   LOCATE 2,20:PRINT"Ansprechpartner:"
460   PEN 3
470   LOCATE 2,22:PRINT I$;"        Daten ok? j=ja   n=nein                 "; I$
480   PEN 2
490   REM
500   REM   - - - - Eingabefelder mit Punkten markieren - - -
510   REM
520   FOR I=6 TO 20 STEP 2
530   LOCATE 19,I
540   PRINT STRING$(21,".")
550   NEXT I
560   REM
570   REM   - - - - Textcursor für Eingabe positionieren - -
580   REM
590   PEN 3
600   LOCATE 19,6:INPUT"",KN
610   LOCATE 19,8:INPUT"",KN$
620   LOCATE 19,10:INPUT"",S$
630   LOCATE 19,12:INPUT"",O$
640   LOCATE 19,14:INPUT"",T$
650   LOCATE 19,16:INPUT"",WK$
660   LOCATE 19,18:INPUT"",UMSATZ
670   LOCATE 19,20:INPUT"",ANSP$
680   A$=INKEY$:IFA$=""THEN680
690   IF A$="j" THEN 470
700   IF A$="n" THEN GOTO 100
710   SOUND 1,300,100,7
720   SOUND 2,301,100,7
730   GOTO 680
740   REM   - - - - weiter im Programm ........ - - - -
```

Programmbeschreibung:

Zeile 100 bis 140: Bildschirmmodus wählen und Farben einstellen.

Zeile 150: Bei Verwendung von I$ in einer PRINT-Anweisung wird alles nach I$ revers ausgegeben. Beim nächsten I$ wird der Reversmodus wieder abgeschaltet.

Zeile 160 bis 180: Oberen Balken aus Grafikzeichen zusammensetzen. Unteren Balken aus Grafikzeichen zusammensetzen. Seitlichen Balken definieren.

Zeile 190 bis 340: Den gesamten Rahmen zeichnen.

Zeile 350: Auf andere Zeichenfarbe umschalten.
Zeile 360 bis 450: Textcursor positionieren und die jeweiligen Bezeichnungen an vorbestimmter Stelle ausgeben.
Zeile 460: Zeichenfarbe ändern.
Zeile 470: Text ausgeben, ob die eingegebenen Daten so richtig sind. Dieser Text erscheint in revers.
Zeile 480: Zeichenfarbe wieder umschalten.
Zeile 490 bis 510: Kommentarzeilen
Zeile 520 bis 550: Eingabefelder mit Punkten markieren.
Zeile 560 bis 580: Kommentarzeilen
Zeile 590: Zeichenfarbe wieder wechseln.
Zeile 600 bis 670: Textcursor positionieren und mit INPUT Daten eingeben. Die Darstellung INPUT"",Variable verhindert das Erscheinen des Fragezeichens innerhalb dieser Maske.
Zeile 680: Abfrage, ob eine Taste auf der Tastatur gedrückt wurde.
Zeile 690: Wenn die Taste "j" gedrückt wird, dann nach Zeile 740 gehen und dort im Programm fortfahren.
Zeile 700: Wenn die Taste "n" gedrückt wird, dann die gesamte Eingabe wiederholen.
Zeile 710 und 720: Einen zweistimmigen Ton als Warnsignal für eine falsche Eingabe erzeugen.
Zeile 730: Sprung zurück zur Tastaturabfrage nach Zeile 680.
Zeile 740: Hier gehts im Programm später weiter.

Tips zum Programm:

Sie haben nun eine recht übersichtliche Eingabemaske für die Erfassung von Daten kennengelernt. Damit dürfte es auch möglich sein, jemanden vor den Computer zu setzen, der von der Technik und vom Programmieren im Prinzip keine Ahnung hat. Beispielsweise könnte in einem Betrieb auch jemand, der keine EDV-Kenntnisse hat, diese Daten über diese Maske erfassen. Voraussetzung dafür ist natürlich eine übersichtliche Bildschirmgestaltung, die leider auch heute noch bei den vielen Möglichkeiten, die einem zur Gestaltung zur Verfügung stehen, recht stiefmütterlich behandelt wird. Gerade bei Programmen im Bereich der Personal Computer kommt die Benutzerführung über den Bildschirm viel zu kurz. Da vielen Endanwendern das jeweilige Handbuch meist sowieso zu kompliziert erscheint, wird die Leistungsfähigkeit der Software nur teilweise ausgeschöpft. Es sollte also darauf geachtet werden, daß der Endanwender möglichst unkompliziert über den Bildschirm durch das Programm geführt wird und er auf das Handbuch nur bei speziellen Problemen zurückgreifen muß.

Diese Eingabemaske können Sie natürlich auch wieder nach Belieben ändern und weiter ausbauen. Für neue Begriffe brauchen Sie nur die Bezeichnungen der Zeilen 380 bis 450 sowie die nötigen Variablen in den Zeilen 600 bis 670 auszutauschen. Falls Sie mehrere Eingabefelder wünschen, müßten Sie die Endwerte der Schleifen um einige Positionen höher setzen.

Eine weitere Möglichkeit ist die Begrenzung der Eingabe auf eine bestimmte Anzahl Zeichen. Wenn Sie ein wenig Erfahrung im Programmieren haben, wird auch dies für Sie kein Problem mehr sein.

Im nächsten Beispiel wollen wir eine Figur auf dem Bildschirm darstellen.

14.3 Roboter

Bei der Vielzahl der Grafikzeichen bietet sich die Programmierung von Bildern geradezu an. In diesem Beispiel zeige ich Ihnen einen kleinen Roboter, der sich aus lauter Grafikzeichen zusammensetzt. Für solche Darstellungen auch hier wieder ein Tip: Verwenden Sie die Bildschirmarbeitsblätter im Anhang Ihres Schneider-Handbuchs. Und nun zu dem versprochenen Programm:

```
100   MODE 1
110   CLS
120   INK 0,0
130   BORDER 0
140   PEN 2
150   REM  - - - - Kopf zeichnen - - - -
160   REM
170   PRINT TAB(15);CHR$(240);TAB(24);CHR$(240)
180   PRINT TAB(12);CHR$(194);STRING$(2,CHR$(154));CHR$(155);
      STRING$(8,CHR$(154));CHR$(155);STRING$(2,CHR$(154));CHR$(195)
190   FOR I= 1 TO 5
200   PRINT TAB(12);CHR$(149);TAB(27);CHR$(149)
210   NEXT
220   PRINT TAB(12);CHR$(147);STRING$(14,CHR$(154));CHR$(153)
230   REM  - - - - Kopf fertig gezeichnet - - - -
240   REM
250   REM
260   REM
270   PEN 1
280   REM  - - - - Körper zeichnen - - - -
290   REM
300   PRINT TAB(16);CHR$(149);STRING$(6,CHR$(32));CHR$(149)
310   PRINT TAB(16);CHR$(149);STRING$(6,CHR$(32));CHR$(149)
320   PRINT TAB(10);CHR$(204);STRING$(18,CHR$(208));CHR$(205)
330   FOR I=1 TO 7
340   PRINT TAB(10);CHR$(211);STRING$(18,CHR$(32));CHR$(209)
```

```
350   NEXT
360   PRINT TAB(10);CHR$(139);STRING$(18,CHR$(140));CHR$(135)
370   REM   - - - - Körper fertig gezeichnet - - - -
380   REM
390   REM
400   REM   - - - - Beine und Füße zeichnen - - - -
410   REM
420   PRINT TAB(12);CHR$(204);STRING$(2,CHR$(208));CHR$(205)
430   PRINT TAB(24);CHR$(204);STRING$(2,CHR$(208));CHR$(205)
440   FOR P=1 TO 3
450   PRINT TAB(12);CHR$(211);STRING$(2,32);CHR$(209);TAB(24);CHR$(211);
      STRING$(2,32);CHR$(209)
460   NEXT
470   PRINT TAB(11);CHR$(204);STRING$(4,CHR$(208));CHR$(205);
480   PRINT TAB(23);CHR$(204);STRING$(4,CHR$(208));CHR$(205);
490   PRINT TAB(11);CHR$(204);STRING$(4,CHR$(210));CHR$(205);
500   PRINT TAB(23);CHR$(204);STRING$(4,CHR$(210));CHR$(205);
510   REM
520   REM   - - - - Beine und Füße sind jetzt fertig - - -
530   REM
540   REM
550   REM   - - - - Gesicht zeichnen - - - -
560   REM
570   PEN 3
580   LOCATE 15,4:PRINT CHR$(230);STRING$(8,32);CHR$(230)
590   LOCATE 18,7:PRINT TAB(137);CHR$(140);CHR$(140);CHR$(134)
600   FOR I=1 TO 2000:NEXT
610   FOR I=1 TO 15
620   LOCATE 15,4: PRINT CHR$(231):FOR T=1 TO 200:NEXT
630   SOUND 1,200,10,7
640   SOUND 2,202,10,7
650   LOCATE 15,4:PRINT CHR$(230):FOR T=1 TO 100:NEXT
660   NEXT 1
670   PEN 1
680   REM
690   REM
700   REM   - - - - Kleinen Bildschirm (Window) definieren -
710   REM
720   WINDOW#1,12,27,12,17
730   PAPER#1,2
740   CLS#1
750   PEN#1,0
760   PRINT#1:PRINT#1
770   PRINT#1,"    Gleich gehts"
780   PRINT#1,"          los"
790   FOR I=200 TO 100 STEP -10
800   FOR T=1 TO 300:NEXT T
810   SOUND 1,I,10,7
820   SOUND 2,I,-1,10,7
830   LOCATE 18,7:PRINT CHR$(137);CHR$(140);CHR$(140);CHR$(134)
840   FOR T=1 TO 300:NEXT T
850   LOCATE 18,7:PRINT STRING$(4,CHR$(140))
860   NEXT
870   LIST#1
```

Programmbeschreibung:

Im Prinzip dokumentiert sich das Programm weitgehend selbst. Ich möchte nur die Bewegungen und das Listing für den Körper noch ein wenig erläutern. Zu den Bewegungen: Das Auge blinkt, weil zeitlich versetzt ein gleiches, aber ausgefülltes Zeichen an dieselbe Position gebracht wird. Der Mund scheint sich nur zu bewegen, weil damit ebenso verfahren wird, wofür die Zeilen 830 bis 850 verantwortlich sind. Durch kleine Änderungen lassen sich auch andere Körperteile, wie etwa die Füße, leicht bewegen.

Im nächsten Beispiel möchte ich kurz auf Bewegungen auf dem Bildschirm eingehen und an kleinen Demonstrationsprogrammen zeigen, wie man diese programmieren kann.

14.4 Programmieren von Bildschirmbewegungen

Sie wissen bereits, wie sich einzelne Teile bewegen lassen. In diesem Abschnitt geht es nun darum, wie man Figuren über den gesamten Bildschirm laufen lassen kann. Unter den letzten Zeichen im Zeichensatz des Schneiders befinden sich mehrere Männchen, die sich für solche Beispiele recht gut verwenden lassen. Sehen wir uns dies einmal näher an:

```
10    MODE 1
20    CLS
30    INK 0,0
40    BORDER 0
50    FOR I=1 TO 40
60    LOCATE I,24
70    PRINT CHR$(250)
80    FOR T=1 TO 200:NEXT T
90    LOCATE I,24
100   PRINT CHR$(251)
110   FOR T=1 TO 100:NEXT T
120   LOCATE I,24
130   PRINT" "
140   NEXT I
150   RUN
```

Programmbeschreibung:

Die Zeilen 10 bis 40 stellen wieder Bildschirmformat und Farben ein. Da das Männchen unten über den Bildschirm laufen soll und der Modus 1 insgesamt 40 mögliche Zeichen pro Zeile zur Verfügung stellt, lautet die Schleife dafür `FOR I=1 TO 40`.

Anschließend wird in Zeile 60 die ermittelte Position in Zeile 24 auf dem Bildschirm angesteuert.

In Zeile 70 wird dann das Männchen mit der Code-Nr. 250 ausgegeben. Damit nicht alles so schnell geht, habe ich noch eine kleine Warteschleife in den Zeilen 80 und 110 installiert. Nachdem die Warteschleife vom Rechner abgearbeitet wurde, wird das Männchen mit der Code-Nr. 251 auf den Bildschirm gebracht und zwar an derselben Position, an der sich eben noch Männchen-Nr. 250 befand.

Zeile 130 dient dazu, daß das letzte Männchen wieder auf dieser Position gelöscht wird, weil nur so eine echte Bewegung entsteht.

Mit diesem Programm können Sie ausprobieren, welche Zeile für welche Aufgabe zuständig ist. So können Sie beispielsweise die Geschwindigkeit des Ablaufs ändern, indem Sie die Warteschleifen kleiner machen oder vielleicht ganz aus dem Programm nehmen, oder Sie können indem Sie Zeile 130 löschen und anschließend das Programm nochmals starten, herausfinden, wie die scheinbare Bewegung entsteht.

Das zweite Beispiel zeigt ein Männchen, das mit einem Fallschirm vom Himmel fällt und, am Boden angekommen, in ein Haus läuft. Anschließend beginnt das Programm wieder von vorne. In diesem Beispiel werden vertikale und horizontale Bildschirmbewegungen erzeugt.

```
10    MODE 1
20    CLS
30    INK 0,0
40    BORDER 0
50    LOCATE 32,24
60    PRINT CHR$(141)+CHR$(140)+CHR$(140)+CHR$(142)
70    LOCATE 32,21
80    PRINT CHR$(204)+CHR$(208)+CHR$(208)+CHR$(205)
90    LOCATE 32,22:PRINT CHR$(133)+"  "+CHR$(138)
100   LOCATE 32,23:PRINT CHR$(133)+"  "+CHR$(138)
110   FOR I=1 TO 23
120   LOCATE 10,I
130   PRINT CHR$(253)
140   LOCATE 10,I+1
150   PRINT CHR$(248)
160   FOR T=1 TO 250:NEXT T
170   LOCATE 10,I:PRINT" "
180   LOCATE 10,I+1:PRINT" "
190   NEXT I
200   REM
210   REM  - - - - - gelandet - - - -
220   REM
230   FOR I=10 TO 31
240   LOCATE I;24
250   PRINT CHR$(250)
260   FOR T=1 TO 200:NEXT T
```

```
270    LOCATE I,24
280    PRINT CHR$(251)
290    FOR T=1 TO 100:NEXT T
300    LOCATE I,24:PRINT" "
310    NEXT I
320    LOCATE 34,23
330    PRINT CHR$(250)
340    FOR T=1 TO 1000:NEXT T
350    RUN
```

Kurz zur Beschreibung des Programms:

Wichtig ist bei solchen Bewegungen immer, daß Sie die letzte Bildschirmposition auch wieder löschen, wenn Sie eine Position weiter gehen. Der gesamte Ablauf kann beschleunigt werden, indem die Schleifen kürzere Laufzeiten erhalten.

Im nun folgenden Beispiel möchte ich Ihnen zeigen, wie man Texte oder Buchstaben auf dem Bildschirm bewegt. Auch hier sind der eigenen Phantasie keine Grenzen gesetzt.

```
10     MODE 1
20     CLS
30     BORDER 0
40     INK 0,0
50     CALL &BD19
60     FOR I=1 TO 12
70     LOCATE I,I
80     PEN 3
90     PRINT "SCH";
100    PEN 2
110    PRINT"NE";
120    FOR T=1 TO 100:NEXT T
130    IF I<12 THEN LOCATE I,I:PRINT STRING$(5,32)
140    NEXT I
150    FOR I=24 TO 12 STEP -1
160    LOCATE I+5,I
170    PEN 2
180    PRINT"I";
190    PEN 3
200    PRINT"DER"
210    PEN 1
220    FOR T=1 TO 100:NEXT T
230    IF I>12 THEN LOCATE I+5,I:PRINT STRING$(5,32)
240    NEXT I
250    FOR T=1 TO 900:NEXT T
260    FOR I=1 TO 12
270    LOCATE 22,I
280    PRINT"IST"
290    FOR T=1 TO 100:NEXT T
300    LOCATE 22,I
310    IF I<12 THEN PRINT STRING$(3,32)
320    NEXT I
330    FOR I=24 TO 12 STEP -1
```

```
340   LOCATE 26,I
350   PEN 1:PRINT"S";
360   PEN 2:PRINT"P";
370   PEN 3:PRINT"I";
380   PEN 1:PRINT"T";
390   PEN 2:PRINT"Z";
400   PEN 2:PRINT"E";
410   FOR T=1 TO 100:NEXT T
420   LOCATE 26,I
430   IF I>12 THEN PRINT STRING$(6,32)
440   NEXT I
450   PEN 1
460   FOR T=1 TO 1000:NEXT T
470   FOR I=39 TO 33 STEP -1
480   LOCATE I,12
490   PRINT"!!"
500   FOR T=1 TO 100:NEXT T
510   IF I>33 THEN LOCATE I,12:PRINT STRING$(3,32)
520   NEXT I
530   FOR T=1 TO 1000:NEXT T
540   FOR I=1 TO 30
550   PRINT
560   FOR T=1 TO 200:NEXT T
570   NEXT I
580   RUN
```

Kurz zur Beschreibung des Programms:

In Zeile 50 finden Sie einen CALL-Aufruf, der dafür sorgt, daß die einzelnen Phasen der Bewegungen nicht so abgehackt erscheinen. Innerhalb jeder Schleife zur Textverschiebung befindet sich eine Abfrage mit IF...THEN. Diese Abfrage hat dafür zu sorgen, daß beim letzten Durchlauf der Schleife der Text nicht gelöscht wird, sondern dort stehen bleibt.

Nutzen Sie dieses Programm für weitere Experimente, indem Sie beispielsweise Geschwindigkeit und Positionen beliebig variieren.

15 Alternative Zeichensätze

Wie im letzten Kapitel bereits erwähnt wurde, gibt es beim CPC 464 die Möglichkeit, selbst beliebige Zeichen zu erstellen und diese gegen bereits vorhandene auszutauschen. In erster Linie ist dies natürlich für den deutschen Zeichensatz interessant, da er im Standardzeichensatz fehlt. Aber auch Sonderzeichen wie das Quadratzeichen, können so definiert werden.

Leider ist die ganze Handhabung für einen Anfänger jedoch recht umständlich, da mit Bits und Bytes sowie dem dualen und dezimalen Zahlensystem gearbeitet wird. Ich möchte Ihnen daher zuerst kurz die Zusammenhänge für den Aufbau eines Zeichens erklären und dann eine recht einfache Methode darlegen, mit der man sich selbst Sonderzeichen erstellen kann.

Betrachten wir uns dazu einmal ein Zeichen. Ein Zeichen besteht aus insgesamt 64 verschiedenen Positionen, die entweder gefüllt oder nicht gefüllt sein können. Das Zeichen mit dem ASCII-Code 32 ist beispielsweise nicht gefüllt, während das Zeichen mit der Nummer 143 voll gefüllt ist. Wenn wir uns einen Buchstaben vornehmen und diesen genau untersuchen, erkennen wir auch hier, daß er aus lauter Punkten zusammengesetzt ist.

Die Grafik, die ich Ihnen jetzt vorstellen möchte, macht dies noch etwas deutlicher.

Bytes	Bits 7	6	5	4	3	2	1	0	Dezimal-werte
Byte 1		●							64
Byte 2		●							64
Byte 3		●							64
Byte 4		●							64
Byte 5		●							64
Byte 6		●	●						96
Byte 7		●	●	●	●	●	●	●	127
Byte 8									0

Bild 15.1: *Beispiel eines Buchstabens aus dem Zeichensatz, hier der Buchstabe L*

Sie erkennen schon an dieser Darstellung, daß sich ein Zeichen aus insgesamt 8 Bytes zusammensetzt, wobei jedes Byte aus 8 einzelnen Bits besteht. Interessant ist für uns in erster Linie der Aufbau der einzelnen Bits. Sie bestimmen nämlich, welcher Punkt auf dem Bildschirm gesetzt wird und welcher nicht. Dabei wird nach einem relativ einfachen Prinzip gearbeitet: Soll ein Punkt an einer Stelle später erscheinen, dann setzen Sie das entsprechende Bit auf 1. Der Computer kennt nur zwei Zustände; entweder der Punkt leuchtet später, dann bekommt er eine 1 oder der Punkt bleibt dunkel, dann bekommt er eine 0 zugeteilt.

Bis jetzt leuchten unsere Punkte jedoch noch nicht. Wir müssen dem Computer nämlich noch mitteilen, wie unser Zeichen später aussehen soll. Dies geschieht mit Hilfe von Zahlen, die nach einem speziellen BASIC-Befehl erscheinen.

Bleiben wir bei unserem Buchstaben L, den wir nun entwerfen wollen. Als erstes müssen wir dem System mitteilen, wie die Nullen und Einsen verarbeitet werden sollen. Damit haben wir das sogenannte Bitmuster festgelegt, d.h. welcher Punkt leuchten soll und welcher nicht.

Im zweiten Schritt erfolgt nun die Umrechnung in für den Computer verständliche Werte. Dabei muß zuerst für jedes Bit innerhalb eines Bytes

die passende Dezimalzahl ermittelt werden. Während wir in unserem dezimalen Zahlensystem als Basis die 10 haben, hat das binäre Zahlensystem, mit dem unser Computer arbeitet, als Basis die 2.

Wenn wir uns das erste Byte für unseren Buchstaben L ansehen, dann ergibt sich daraus folgendes Bitmuster:

Bit	7	6	5	4	3	2	1	0
	0	1	0	0	0	0	0	0

Für dieses Bitmuster des ersten Bytes muß jetzt der Dezimalwert ermittelt werden. Dabei geht man so vor:

Bit 7:	wenn auf 1 dann 1*2*2*2*2*2*2*2	$=2^7$	ergibt dezimal 128
Bit 6:	wenn auf 1 dann 1*2*2*2*2*2*2	$=2^6$	ergibt dezimal 64
Bit 5:	wenn auf 1 dann 1*2*2*2*2*2	$=2^5$	ergibt dezimal 32
Bit 4:	wenn auf 1 dann 1*2*2*2*2	$=2^4$	ergibt dezimal 16
Bit 3:	wenn auf 1 dann 1*2*2*2	$=2^3$	ergibt dezimal 8
Bit 2:	wenn auf 1 dann 1*2*2	$=2^2$	ergibt dezimal 4
Bit 1:	wenn auf 1 dann 1*2*1.	$=2^1$	ergibt dezimal 2
Bit 0:	wenn auf 1 dann 1*2*0	$=2^0$	ergibt dezimal 1

Anschließend werden alle Werte für ein Byte addiert, deren Bits auf 1 stehen. Für unser L bedeutet dies für das erste Byte:

```
Bit 7: nicht gesetzt, daher 0
Bit 6: gesetzt, daher      64
Bit 5: nicht gesetzt, daher 0
Bit 4: nicht gesetzt, daher 0
Bit 3: nicht gesetzt, daher 0
Bit 2: nicht gesetzt, daher 0
Bit 1: nicht gesetzt, daher 0
Bit 0: nicht gesetzt, daher 0
-----------------------------
Addition aller Werte       64
```

Dieser Wert erscheint dann rechts in der Darstellung unter der Spalte Dezimalwert. Er wird später im BASIC-Befehl verwendet.

Bisher haben wir uns jedoch nur das erste Byte von insgesamt acht Bytes angesehen. Da die Bytes drei, vier und fünf ebenso aussehen wie Byte eins, ergibt sich dafür auch der gleiche Dezimalwert. Bei Byte 6 sieht es schon wieder etwas anders aus. Hier sind zwei Bits gesetzt. Der Dezimalwert für Byte sechs wird dann so ermittelt:

```
Bit 7: nicht gesetzt, daher 0
Bit 6: gesetzt, daher      64
Bit 5: gesetzt, daher      32
Bit 4: nicht gesetzt, daher 0
Bit 3: nicht gesetzt, daher 0
Bit 2: nicht gesetzt, daher 0
Bit 1: nicht gesetzt, daher 0
Bit 0: nicht gesetzt, daher 0
-----------------------------
Addition aller Werte       96
```

Ebenso wird Byte sieben nach diesem Schema berechnet:

```
Bit 7: nicht gesetzt, daher 0
Bit 6: gesetzt, daher      64
Bit 5: gesetzt, daher      32
Bit 4: gesetzt, daher      16
Bit 3: gesetzt, daher       8
Bit 2: gesetzt, daher       4
Bit 1: gesetzt, daher       2
Bit 0: gesetzt, daher       1
-----------------------------
Addition aller Werte      127
```

Da das achte Byte komplett leer ist, erhält es den dezimalen Wert 0.

Wir haben jetzt das Bitmuster in dezimale Werte umgewandelt und die Zahlen 64, 64, 64, 64, 64, 96, 127, 0 erhalten. Nun muß der Rechner wieder arbeiten. Zum Umdefinieren gibt es den Befehl SYMBOL. Er hat folgendes Format:

```
SYMBOL ASCII-Code, Dezimalzahlen 1 bis 8
```

Unser Symbol-Befehl für den Großbuchstaben L sieht demnach so aus:

```
SYMBOL 32,64,64,64,64,64,96,127,0
```

Da der Zeichensatz aus nur 255 verschiedenen Zeichen bestehen kann und wir jetzt ein Zeichen mehr haben, müssen wir uns von irgendeinem Zeichen trennen; an dessen Stelle tritt dann unser Großbuchstabe L. Die erste Angabe nach dem SYMBOL-Befehl sagt in unserem Beispiel, daß das neue Zeichen auf die Leertaste gelegt wird. Da diese den ASCII-Code 32 hat, steht die 32 am Anfang der Zahlenkolonne.

Nun muß dem Rechner noch mitgeteilt werden, ab welchem ASCII-Code neu definiert werden soll. Dies geschieht mit

```
SYMBOL AFTER 32
```

vor dem SYMBOL-Befehl. Unser kleines Beispielprogramm könnte vielleicht so aussehen:

```
10    MODE 0
```

```
20   CLS
30   SYMBOL AFTER 32
40   SYMBOL 32,64,64,64,64,64,96,127,0
50   PRINT"Normales L ";" neues: ";CHR$(32)
```

Normalerweise ist der SYMBOL AFTER-Befehl eingestellt auf 240. Das bedeutet, wenn Sie ab ASCII-Code 240 umdefinieren wollen, brauchen Sie dies nicht extra mitzuteilen. Aber Vorsicht mit dem Befehl SYMBOL AFTER! Hier werden nämlich alle vorher definierten Zeichen wieder gelöscht. Deshalb empfehle ich Ihnen, die Anweisung SYMBOL AFTER ganz an den Anfang des Programms zu stellen, damit sie nur einmal ausgeführt wird.

Auf den nächsten Seiten folgen einige Beispiele für den deutschen Zeichensatz und einige nützliche Sonderzeichen, die Sie sicher recht gut in eigenen Programmen verwenden können.

15.1 Erstellung eines Sonderzeichens

Im letzten Abschnitt haben Sie erfahren, wie Sie einzelne Zeichen neu erstellen können. Im vorliegenden Abschnitt werden Ihre bisherigen Kenntnisse noch weiter vertieft und anhand eines Sonderzeichens, das sicherlich jeder gebrauchen kann, wird der gesamte Vorgang von der Planung bis zum fertigen Zeichen nochmals am Beispiel aufgezeigt.

Das Sonderzeichen, das hier erstellt werden soll, ist ein Telefonapparat, wie Sie ihn beispielsweise aus Zeitungsanzeigen kennen.

Noch einmal die Arbeitsweise in einzelnen Schritten:

1. Hier sind Ihre kreativen Fähigkeiten gefragt. Sie erstellen sich mit Hilfe einer acht-mal-acht- Kästchen großen Matrix Ihr Zeichen.

2. Für jedes Bit muß nun der Dezimalwert ermittelt werden. Die Summe aller Werte eines Bytes ergibt den nötigen Wert für den Befehl SYMBOL.

3. Alle nötigen Dezimalwerte hinter den Befehl SYMBOL setzen und evtl. vorher mit SYMBOL AFTER den Änderungsbereich festlegen.

4. Falls gewünscht, dieses Zeichen einer Taste zuweisen.

Auf der nächsten Seite finden Sie einen Entwurf in der acht-mal-acht-Kästchen großen Matrix. Durch einfaches Markieren wird das Bitmuster

für jedes Byte festgelegt. Anschließend erfolgt die Umrechnung, die dann nochmals näher erläutert wird.

Bytes	Bits 7	6	5	4	3	2	1	0	Dezimal-werte
Byte 1		●	●	●	●	●	●		126
Byte 2	●	●	●	●	●	●	●	●	255
Byte 3	●	●					●	●	195
Byte 4				●	●				24
Byte 5		●	●	●	●	●	●		126
Byte 6	●	●	●	●	●	●	●	●	255
Byte 7	●	●	●	●	●	●	●	●	255
Byte 8	●	●	●	●	●	●	●	●	255
Bitmuster des Telefonzeichens									

Bild 15.2: *Bitmuster des Telefonzeichens*

Und nun noch einmal die Berechnung:

Byte 1:

Bit-Nr.	gesetzt	nicht gesetzt	Dezimalwert
7		X	0
6	X		64
5	X		32
4	X		16
3	X		8
2	X		4
1	X		2
0		X	0
Addition der Dezimalwerte:			126

Byte 2: Alle Bits gesetzt. Dies ergibt den Dezimalwert von 255.

Byte 3:

Bit-Nr.	gesetzt	nicht gesetzt	Dezimalwert
7	X		128
6	X		64
5		X	0
4		X	0
3		X	0
2		X	0
1	X		2
0	X		1
Addition der Dezimalwerte:			195

Byte 4:

Bit-Nr.	gesetzt	nicht gesetzt	Dezimalwert
7		X	0
6		X	0
5		X	0
4	X		16
3	X		8
2		X	0
1		X	0
0		X	0
Addition der Dezimalwerte:			24

Byte 5: Identisch mit Byte 1, also ergibt sich der Dezimalwert 126.

Byte 6 bis Byte 8: Alle Bits gesetzt, ergibt also 255.

Damit ist der zweite Schritt, d.h. die Berechnung der dezimalen Werte abgeschlossen. Im dritten Schritt wird das neue Zeichen definiert. Möchten Sie es z.B. auf ASCII-Code 240 legen, dann sieht der Befehl dafür so aus:

```
SYMBOL 240,126,255,195,24,126,255,255,255
```

Schalten Sie jetzt in den `MODE 0` und betrachten Sie sich mit `PRINT CHR$(240)` Ihr neues Zeichen.

Nach diesem Verfahren lassen sich alle möglichen Zeichen erzeugen. Sie sollten nach Durcharbeit dieses Kapitels in der Lage sein, selbst beliebige Zeichen zu entwerfen.

15.2 Erstellen der deutschen Zeichen

In diesem Abschnitt geht es um die Erstellung der deutschen Zeichen, wie Sie sie von der Schreibmaschine her gewohnt sind. Ich selbst arbeite normalerweise mit einer Schreibmaschine vom Typ Olympia, die ich an den Computer angeschlossen habe und von der ich die Vorlage der Tastatur übernommen habe. Da ich mich nicht umgewöhnen wollte, habe ich die gleiche Tastaturbelegung wie bei meiner Schreibmaschine gewählt. Doch sehen wir uns zunächst einmal an, was wir überhaupt zu tun beabsichtigen.

1. Das ß soll auf die Taste mit dem Hochpfeil gelegt werden.
2. Die Buchstaben Z und Y sollen auf der Tastatur vertauscht werden. Das gilt für die Groß- und Kleinbuchstaben.
3. Das kleine und das große Ö sollen auf die Tastatur gelegt werden.
4. Das kleine und das große Ü sollen auf die Tastatur gelegt werden.
5. Das kleine und das große Ä sollen auf die Tastatur gelegt werden.
6. Zusätzlich soll ein Telefonzeichen erstellt werden.

Wie geht man zweckmäßigerweise dabei vor:

1) Alle Symbolanweisungen erstellen
2) Alle Symbolanweisungen auf bestimmte Tasten legen
3) Programm mit RUN aktivieren und anschließend löschen.

Im ersten Schritt wollen wir alle Symbolanweisungen generieren. Ich habe dafür ein bereits fertig erarbeitetes Programm zur Erstellung von beliebigen Zeichen verwendet, was die spätere Umrechnung in die dezimalen Werte erheblich vereinfacht. Solche Programme werden in Computerzeitschriften und in einigen Büchern lauffertig angeboten. Sie brauchen das Programm dann nur noch abzutippen und können so auf recht einfache Art ein neues Zeichen auf dem Bildschirm kreieren.

Anschließend berechnen solche Programme die Dezimalwerte für jedes Byte und geben sie auf dem Bildschirm aus. Es ergibt sich somit ein erheblicher Zeitvorteil gegenüber der herkömmlichen Verfahrensweise. Sie sollten aber mit solchen Zeichensatzgeneratoren erst arbeiten, wenn Sie das Prinzip der Berechnung verstanden und es auch einmal ausprobiert haben.

Nun die mit dem Zeichensatzgenerator ermittelten Zeichen:

Befehl	**Buchstabe**
SYMBOL 91,195,60,102,102,126,102,102,0	Ä
SYMBOL 92,195,60,102,102,102,102,60,0	Ö
SYMBOL 93,102,0,102,102,102,102,60,0	Ü
SYMBOL 123,108,0,120,12,124,204,118,0	ä
SYMBOL 124,102,0,60,102,102,102,60,0	ö
SYMBOL 125,102,0,102,102,102,102,62,0	ü
SYMBOL 126,60,102,102,108,102,102,110,96	ß
SYMBOL 130,126,255,195,24,60,126,126,126	Telefon-zeichen

Das wäre also der erste Schritt, die Erstellung der Sonderzeichen. Jetzt müssen diese Sonderzeichen an genau definierte Stellen auf die Tastatur gelegt werden. Dazu benötigen Sie die Tastaturnummern der einzelnen Tasten. Eine Aufstellung über die Nummern finden Sie im Anhang dieses Buches. Man ordnet nun der jeweiligen Tastaturnummer mit dem Befehl KEY DEF das neue Zeichen zu. Und so wendet man diesen Befehl an:

```
KEY DEF Tasten-Nr., Zeichenwiederholung,
Zeich.-Normal, Zeich.- SHIFT, Zeich.-CTRL
```

Den Befehl kennen Sie nun zwar, aber vermutlich sind Ihnen die einzelnen Angaben noch fremd. Beginnen wir daher mit der Erläuterung der Zeichenwiederholung: Steht später an dieser Stelle eine 1, so wiederholt sich dieses Zeichen bei längerem Tastendruck immer wieder. Sollten Sie statt einer 1 eine 0 einfügen, ist die Wiederholfunktion außer Kraft gesetzt. In diesem Fall erscheint dann auf Tastendruck nur noch jeweils ein Zeichen auf dem Bildschirm.

Die nächste Angabe gibt an, welches Zeichen erscheint, wenn man die Taste normal drückt, als ohne <SHIFT> und <CTRL>. Es folgt die Angabe, welches Zeichen erscheinen soll, wenn die Tasten <SHIFT> und die jeweilige Taste gedrückt werden. Zum Schluß kommt dann die Angabe, welches Zeichen erscheinen soll, wenn mit der normalen Taste gleichzeitig die <CTRL>-Taste betätigt wird.

Dies ist beispielsweise bei den Buchstaben Ü und ü nützlich. Im normalen Zustand erscheint das kleine ü; drückt man zusätzlich noch die Taste <SHIFT>, kommt der passende Großbuchstabe auf den Bildschirm. Für unser Beispiel bedeutet das:

Die Zeichen [,] und @ werden umdefiniert und mit den Buchstaben Ü, Ö, Ä, ü, ö und ä belegt.

Die Taste [hat die Tastaturnummer 17.

Die Taste] hat die Tastaturnummer 19.

Die Taste @ hat die Tastaturnummer 26.

Die dazugehörigen Befehle lauten:

```
KEY DEF 17, 1, 125, 93
KEY DEF 19, 1, 123, 91
KEY DEF 26, 1, 124, 92
```

Jetzt bleibt nur noch die Frage offen, wohin mit dem erstellten ß? Da man sich von dem Hochpfeil auf der Tastatur wohl am ehesten trennen kann, habe ich dieses Zeichen auf den Hochpfeil (^) gelegt, der die Tastaturnummer 24 hat. Dies geschieht mit dem Befehl

```
KEY DEF 24,1,126
```

Jetzt wird also das Zeichen vom ASCII-Code 126 auf die Tastaturnummer 24 gelegt, wobei die Wiederholungsfunktion aktiviert bleibt.

Zu guter Letzt muß nur noch das Problem mit der Vertauschung der Buchstaben Z, z und Y,y gelöst werden. Dazu brauchen wir kein neues Zeichen zu erstellen, sondern nur die einzelnen ASCII-Codes vertauschen. Die Taste Z hat die Tastaturnummer 71 und das Y hat die Tastaturnummer 43. Außerdem haben die Buchstaben Z, z, Y und y folgenden ASCII-Code-Werte, die Sie im Anhang in der Tabelle nachsehen können.

Zeichen	**ASCII-Code**
Z	90
z	122
Y	89
y	121

Unsere Aufgabe besteht jetzt darin, auf die Tastennummer 71 die Zeichen y, Y und auf die Tastennummer 43 die Zeichen z, Z zu legen.

Die dazu nötigen Befehle lauten:

```
KEY DEF 71,1,121,89
KEY DEF 43,1,122,90
```

Sie sehen, die Kleinbuchstaben können ohne <SHIFT> und die Großbuchstaben mit <SHIFT> erreicht werden.

Wichtig für die Belegung der deutschen Zeichen ist eigentlich nur, daß sie nach dem normalen Alphabet stehen sollten. Dies ist später für die Sortierung von Vorteil, da dann nur noch ein Bereich abgefragt werden braucht und nicht einzelne ASCII-Codes ab 240 bis 255.

Ein weiteres Problem ist die Druckeranpassung. Da jeder Drucker andere Zeichen hat, ist dies meist eine recht zeitaufwendige Angelegenheit.

Abschließend nochmals das Programm für den "Deutschen Zeichensatz":

```
100   KEY DEF 71,1,121,89
110   KEY DEF 43,1,122,90
120   KEY DEF 24,1,126
130   KEY DEF 26,1,124,92
140   KEY DEF 17,1,125,93
150   KEY DEF 19,1,123,91
160   SYMBOL AFTER 32
170   SYMBOL 91,195,60,102,102,126,102,102,0
180   SYMBOL 92,195,60,102,102,102,102,60,0
190   SYMBOL 93,102,0,102,102,102,102,60,0
200   SYMBOL 123,108,0,120,12,124,204,118,0
210   SYMBOL 124,102,0,60,102,102,102,62,0
230   SYMBOL 126,60,102,102,108,102,102,110,96
240   SYMBOL 130,126,255,195,24,60,126,126,126
```

Zusammenfassung:

Dieses Programm sorgt dafür, daß Sie wie auf einer Schreibmaschine mit deutschen Zeichen arbeiten können. Neben den Buchstaben Ä, Ö, Ü, ä, ü, ö, ß werden auch die Buchstaben Z und Y auf der Tastatur vertauscht. Außerdem wird im ASCII-Code 130 ein nützliches Sonderzeichen zur Verfügung gestellt. Dieses Sonderzeichen erreichen Sie zukünftig mit `PRINT CHR$(130)`.

16 Arbeiten mit Bildschirmfenstern (WINDOWS)

Sicherlich ist Ihnen der Begriff WINDOW schon in den letzten Programmen aufgefallen. In diesem Kapitel möchte ich Ihnen nun zeigen, wie man Bildschirmfenster aufbaut und was man damit alles machen kann.

Die WINDOW-Technik eröffnet dem Anwender eine Vielzahl an Möglichkeiten auf dem Bildschirm. mit dieser Technik können Sie den großen Bildschirm in maximal acht kleine Bildschirme aufteilen, wobei jeder Bildschirm einzeln ansprechbar ist. Dabei wäre es z.B. denkbar, daß, während in dem einen Bildschirm noch Berechnungen ausgeführt werden, im nächsten Bildschirm schon eine Grafik mit den Umsätzen des Vorjahres erscheint. Ist dann der Computer mit der Berechnung fertig, möchte man sich vielleicht zur Kontrolle nochmals Teile des BASIC-Listings auf dem Bildschirm ansehen. Leider wird dabei der Bildschirm immer nach oben weitergeschoben, so daß von der schönen Grafik bald nichts mehr zu sehen ist. Auch das muß nicht sein! Sie können z.B. das Listing nur auf den unteren fünf Zeilen oder nur oben am Bildschirm an einer beliebigen Stelle ausgeben.

Hier eröffnen sich für den Programmierer ungeahnte Möglichkeiten. Wer häufiger schon größere Mengen an Daten auf dem Bildschirm löschen mußte, weiß selbst, wie lange so etwas dauert. Mit dem Bildschirmfenster ist auch das kein großes Problem mehr. Sie geben nur an, innerhalb welcher Grenzen Sie den Bildschirm löschen wollen und schon übernimmt der Computer diese Arbeit für Sie. Die WINDOWS sind also keine Spielerei, sondern leistungsfähige Befehle, die einem viel Mühe ersparen können.

Zusätzlich lassen sich damit Programme bedienungsfreundlicher gestalten. Auch dies ist sehr wichtig, denn immer mehr programmierunkundige Menschen müssen Arbeiten am Computer ausführen.

16.1 Bildschirmfenster erstellen

Sie haben die Möglichkeit, maximal acht Bildschirmfenster zu erstellen, wobei sich einzelne Fenster überschneiden können, falls dies gewünscht wird. Für die Definition eines Fensters stellt der Rechner folgenden Befehl zur Verfügung:

```
WINDOW#NR,LO, RO, AZ, BZ
```

Die einzelnen Parameter haben folgende Bedeutung:

NR	Aktuelle Nummer des Bilschirmfensters (0-7)
LO	Erste Position des Fensters links oben
RO	Rechte Position rechts oben
AZ	Fenster beginnt in der Zeile ....
BZ	Fenster geht bis zur Zeile .....

Die einzelnen Paramter müssen gültig sein, d.h. wenn Sie Bildschirmnummer 11 eingeben, werden Sie mit Sicherheit die Fehlermeldung "*Improper argument*" bekommen. Ebenfalls unzulässig sind Angaben, die sich außerhalb des Bildschirms befinden. Beispielsweise ist dies der Fall, wenn Sie für die Position links oben eine Null eingeben.

Weiterhin sollten Sie beachten, daß die Fenstergöße auch vom jeweiligen Bildschirmmodus (0, 1 oder 2) abhängig ist. So wird ein Fenster, daß im Modus 1 noch 40 Zeichen lang war und den gesamten Bildschirm von links nach rechts umfaßte, im Modus 2 nur noch die Hälfte des Bildschirms ausmachen, weil jetzt 80 Zeichen dargestellt werden können.

Nun wollen wir diese Kenntnisse in die Praxis umsetzen und ein Fenster definieren. Das neue Bildschirmfenster soll im Modus 1 aufgebaut werden und eine Länge von insgesamt 40 Zeichen haben. Es beginnt also in der linken Ecke auf Position 1 und hört mit Position 40 wieder auf. Das Fenster soll außerdem in der ersten Zeilennummer beginnen und bis einschließlich der fünften Bildschirmzeile gehen.

Die Abbildung soll die einzelnen Koordinaten noch einmal grafisch darstellen.

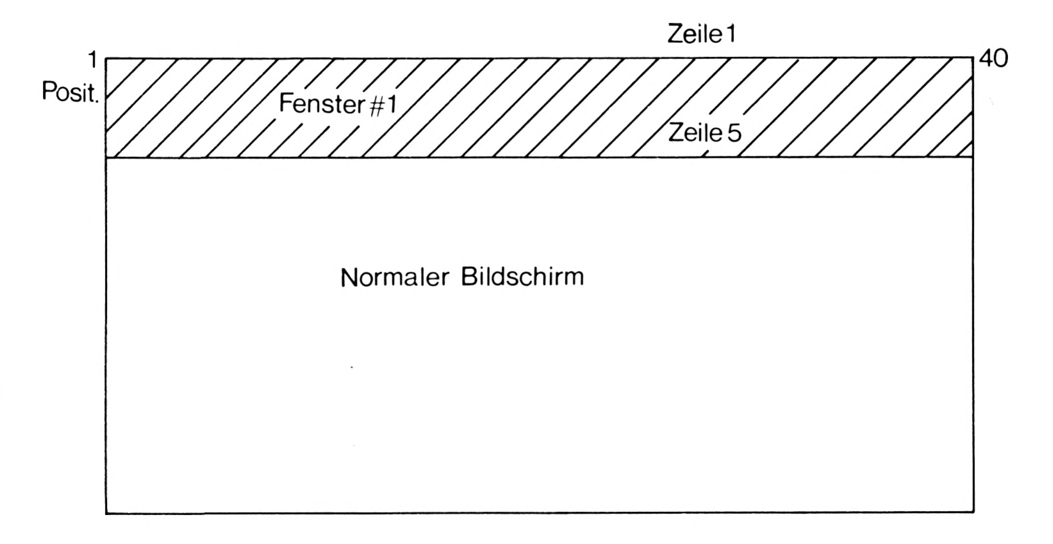


Bild 16.1:

Das neue Fenster soll die Nummer 1 erhalten. Der dazu notwendige Befehl mit allen Parametern lautet:

```
WINDOW#1,1,40,1,5
```

Geben Sie vor dem Befehl noch die Zeilennummer 20 ein. Als Zeile 10 geben Sie bitte MODE 1 ein.

Aktivieren Sie nun Ihr kleines Programm und starten Sie es mit RUN. Es tut sich scheinbar nichts, denn das Programm beendet seine Arbeit mit einem ganz normalen READY. Dennoch haben Sie jetzt ein eigenes Fenster definiert, das Sie nun direkt ansprechen können. Geben Sie zuerst ein:

```
30 CLS#1
```

Diese Zeile bedeutet, daß Sie das erzeugte Bilschirmfenster löschen, falls noch etwas dort stehen sollte. Wie Sie sicherlich bemerkt haben, arbeitet man jetzt mit dem Zusatz #FENSTERNUMMER, um den erstellten Bildschirm anzusteuern.

Alle Befehle, die vorher für den normalen Bildschirm benutzt wurden, lassen sich auch auf dem kleinen Bildschirm anwenden. Dazu gehören u.a. die Befehle:

```
PRINT, CLS, INPUT, LINE, INPUT, PEN, PAPER,
LOCATE
```

und viele mehr.

Sie werden alle mit dem Zusatz #FENSTERNUMMER versehen. Unser Programm sieht jetzt so aus:

```
10 MODE 1
20 WINDOW#1,1,40,1,5
30 CLS#1
```

Um in dieses Fenster einen Text schreiben zu können, geben Sie die folgende Zeile ein.

```
40 PRINT#1,"Dies ist das Fenster-Nr.1"
```

Dazu können Sie noch die drei PRINT-Befehle in Zeile 50 setzen.

```
50 PRINT:PRINT:PRINT
```

Wenn Sie nun das Programm mit RUN starten, sehen Sie oben im Fenster einen Text. Aber ein einwandfreier Beweis für ein Fenster scheint dies immer noch nicht zu sein. Deshalb werden wir im folgenden Schritt das gesamte Fenster umfärben. Damit das Programm jedoch nicht zu schnell abläuft, geben Sie bitte vorher in Zeile 60 die Warteschleife ein.

```
60 FOR I=1 TO 3000:NEXT I
```

Als nächstes muß mit dem schon bekannten Befehl PAPER die Hintergrundfarbe geändert werden. Wenn wir uns dabei für die Farbe rot entscheiden, lautet unsere nächste Zeile:

```
70 PAPER#1,3
```

Nun muß noch mit CLS#1 der gesamte Bereich des Fensters eingefärbt werden. Zeile 80 sieht dann folglich so aus:

```
80 CLS#1
```

Natürlich können Sie Ihr Programm auch auf diesem "kleinen Bildschirm" listen. Dazu geben Sie in Zeile 90 den Befehl LIST#1 ein.

```
90 LIST#1
```

Als letzte Funktion des Programms erscheint das Listing in den oberen fünf Zeilen des Bildschirms. Die Schriftfarbe bleibt gleich, weil sie ja nicht geändert wurde.

16.2 Anwendungsbeispiele mit Bildschirmfenstern

In den folgenden Beispielen wird die WINDOW-Technik noch detaillierter anhand einiger Praxisanwendungen erklärt. Das erste Beispiel beschäftigt sich mit Listings auf dem Bildschirm.

Beispiel 1: Doppeltes Listing

Wer kennt nicht das Problem, daß der Bildschirm manchmal für Programmlistings viel zu klein ist. Da gibt es z.B. zwei Programmteile, die man miteinander vergleichen möchte. Weil diese Programmteile aber recht lang sind, passen sie nicht gleichzeitig auf den Bildschirm. Hier greift nun die WINDOW-Technik helfend ein und teilt den gesamten Bildschirm in drei verschiedene Teilbildshirme ein.

Voraussetzung dafür ist die Arbeit im Bildschirmmodus 2, da hier die 80-Zeichen-Darstellung möglich ist. Der Bildschirm wird dann in der Mitte geteilt, so daß zwei Bildschirme zur Verfügung stehen. Zeile 25 wird über die ganze Bildschirmlänge hinweg als dritter Bildschirm definiert. Auf dieser unteren Zeile werden dann Eingaben und Änderungen vorgenommen. Ich arbeite selbst mit diesem Verfahren und bin damit recht zufrieden, weil gerade bei sehr langen Programmen die Listings übersichtlich verglichen werden können.

Aufgerufen werden diese Listings, indem Sie in Kommandozeile 25 die entsprechenden `LIST`-Befehle eingeben. Wenn Sie beispielsweise den Bereich 1000 bis 2000 und den Bereich 4500 bis 4600 vergleichen wollen, geben Sie unten am Bildschirmrand ein:

```
LIST 1000-2000,#1
```

und drücken die <ENTER>-Taste. Nun kommt noch das 2. Listing auf der rechten Seite:

```
LIST 4500-4600,#2
```

und <ENTER> drücken. Sie sehen jetzt auf der linken Seite den Bereich 1000 bis 2000 und auf der rechten Seite den Bereich 4500 bis 4600.

Außerdem habe ich in das Programm einige Besonderheiten eingebaut, indem ich ein paar Funktionstasten mit sehr nützlichen Befehlen belegt habe. Dabei haben einige Tasten auf dem Zehnerblock folgende Bedeutung bekommen:

Taste 0:	Der Befehl LIST erscheint in der untersten Kommandozeile.
Taste 1:	Wird diese Taste gedrückt, erscheint das gewünschte Programmlisting auf Bildschirm 1, d.h. dem linken Bildschirm.
Taste 2:	Das gewünschte Programmlisting erscheint auf der rechten Seite, also auf Bildschirm 2.
Taste 7:	Diese Taste löscht den geamten linken Bildschirm.
Taste 9:	Diese Taste löscht den geamten rechten Bildschirm.

Sie brauchen also nur noch die gewünschten Zeilennummern angeben, der Rest läuft über die Funktionstasten. Gehen Sie dabei wie folgt vor:

1. Drücken Sie die Taste 0, und Sie ersparen sich das Schreiben des Befehls LIST.

2. Geben Sie jetzt den Bereich der Zeilennummern an, die Sie sehen möchten. Dies ist die einzige Eingabe, die Sie machen müssen.

3. Wählen Sie anschließend, auf welchem Bildschirm Sie das Listing sehen möchten. Wenn Taste 1 gedrückt wird, erscheint das Listing auf der linken Bildschirmseite. Wünschen Sie die Ausgabe auf der rechten Seite, drücken Sie Taste 2.

4. Sie möchten einen oder vielleicht beide Bildschirme wieder löschen? Taste 7 löscht den linken, Taste 9 den rechten Bildschirm.

Nachfolgend das Programmlisting für dieses Hilfsprogramm.

```
10 MODE 2
20 INK 0,0
30 BORDER 0
40 CLS
50 PEN 1
60 KEY 0,"LIST"
70 KEY 1,",#1" + CHR$(13)
80 KEY 2,",#2" + CHR$(13)
90 KEY 7,"CLS#1" + CHR$(13)
100 KEY 9,"CLS#2" + CHR$(13)
110 WINDOW#0,1,80,25,25
120 WINDOW#1,1,40,1,24
130 WINDOW#2,41,80,1,24
140 CLS#0
```

```
150 CLS#1
160 CLS#2
```

Das Programm wird nur einmal mit dem Befehl RUN aktiviert, dann läuft es selbsttätig ab.

Wer lieber im Modus 1 weiterprogrammieren, aber auf diese Darstellungsmöglichkeit nicht verzichten möchte, kann folgendes machen:

Hängen Sie das Hilfsprogramm ans Ende Ihres normalen Programms, aber achten Sie darauf, daß dahinter auch wirklich nichts mehr steht! Definieren Sie nun die kleine <ENTER>-Taste wie folgt:

```
KEY 11,"GOTO 60000"+CHR$(13).
```

Die Zeilennummer 60000 ist jetzt diejenige, in der später obiges Programm beginnt. Jetzt brauchen Sie noch eine Taste, mit der Sie diesen Modus wieder verlassen können. Als günstigste Möglichkeit bietet sich hier die Taste 3 an. Sie ist ebenfalls schnell definiert:

```
KEY 3,"MODE 1:CLS"+CHR$(13)
```

Sie können nun mit den beiden Tasten zwischen den beiden Darstellungen hin- und herschalten, wie es gerade erforderlich ist.

Beispiel 2: WINDOW-MAKER mit Grafikdemo

Mit diesem Programm können Sie beliebige Bildschirmfenster zur Probe anlegen, darstellen, ausgeben und ausdrucken sowie zusätzlich ein Grafikdemo abrufen, welches Sie über die Möglichkeiten dieser Technik informiert. Das Programm ist modular aufgebaut, so daß Sie problemlos einige Teile später entfernen können.

Im Programm taucht ein neuer Befehl auf:

```
WINDOW SWAP
```

Dieser Befehl leitet Systemmeldungen von WINDOW#0 auf einen beliebigen anderen Bildschirm um. Zur Erklärung muß gesagt werden, daß alle Systemmeldungen, wie beispielsweise READY oder BREAK, immer auf WINDOW#0 ausgegeben werden. Hat man dort gerade eine sehr übersichtliche Bildschirmmaske aufgebaut, wird bei einer solchen Meldung die gesamte Arbeit des Bildschirmaufbaus zunichte gemacht. Besser wäre es, wenn man solche Meldungen vom System einfach auf ein anderes Bild-

schirmfenster umleiten könnte. Mit dem Befehl WINDOW SWAP haben Sie diese Möglichkeit. Er wird wie folgt angewandt:

```
WINDOW SWAP 0, Ziel-WINDOW
```

Möchten Sie alle Systemmeldungen zukünftig auf WINDOW#4 ausgeben, geben Sie ein:

```
WINDOW SWAP 0,4
```

Programm

```
1000 MODE 1:CLS:INK 0,0:BORDER 0:i$=CHR$(24)
1010 GOSUB 1110
1020 a$=INKEY$:IF a$="" THEN 1020
1030 z=VAL(a$):IF z<1 OR z>7 THEN GOTO 1020
1040 IF a$="1" THEN GOSUB 1390
1050 IF a$="2" THEN GOSUB 1720
1060 IF a$="3" THEN GOSUB 1840
1070 IF a$="4" THEN GOSUB 1930
1080 IF a$="5" THEN GOSUB 2050
1090 IF a$="7" THEN CLS:END
1100 GOTO 1010
1110 BORDER 0:INK 0,0:INK 1,24:MODE 1:CLS:MOVE 0,0
1120 MODE 1:CLS:MOVE 0,0:DRAW 0,399:DRAW 639,339
1130 DRAW 0,399:DRAW 639,399:DRAW 639,0:MOVE 0,0
1140 DRAW 639,0:i=350:MOVE 0,i:DRAW 639,i
1150 GOSUB 1340:MOVE 100,0:DRAW 100,399-50:LOCATE 2,2
1160 PEN 2:PRINT" WINDOW-MAKER  M E N U E":PEN 1
1170 PEN 3
1180 LOCATE 3,5:PRINT"1":LOCATE 10,5
1190 PRINT"WINDOWS ZUR PROBE ANLEGEN"
1200 LOCATE 3,8:PRINT"2":LOCATE 10,8
1210 PRINT"WINDOW darstellen"
1220 LOCATE 3,11:PRINT"3":LOCATE 10,11
1230 PRINT "WONDOW-Parameter ausgeben"
1240 LOCATE 3,14:PRINT"4":LOCATE 10,14
1250 PRINT"WINDOW-Parameter ausdrucken"
1260 LOCATE 3,17:PRINT"5":LOCATE 10,17
1270 PRINT"WINDOW-DEMO zeigen"
1280 LOCATE 3,20
1290 PRINT"6":LOCATE 10,20:PRINT"- - - - - - "
1300 LOCATE 3,23:PRINT"7":LOCATE 10,23
1310 PRINT"Programm beenden"
1320 PEN 1
1330 GOTO 1380
1340 REM einteilung
1350 FOR i=307 TO 0 STEP -49:MOVE 0,i:DRAW 639,i
1360 NEXT:FOR i=15 TO 0 STEP -1:MOVE 0,i
1370 DRAW 639,i: NEXT:RETURN
1380 RETURN
1390 MODE i:CLS
1400 WINDOW#7,1,40,20,25
1410 INPUT"Linke Position oben ";l
1420 INPUT"Rechte Position oben ";r
```

```
1430 INPUT"Links unten ";liu
1440 INPUT"Rechts untern ";reu
1450 MODE 1:CLS
1460 PRINT"WINDOWS zeigen:"
1470 PAPER#1,3:INK 1,24:WINDOW#1,l,rlliu,reu:DLS#1
1480 e$=INKEY$:IF e$="" THEN 1480
1490 MODE 1
1500 PRINT#7,"Der WINDOW-Befehl für Ihr Programm"
1510 PRINT#7,"hat dann folgendes Format:"
1520 PRINT#7
1530 PRINT#7,i$;" WINDOW# Nummer,";l;",";r;",";
1540 PRINT#7,liu;",";reu;" ";i$
1550 PRINT#7,"So richtig? j=ja  n=nein"
1560 e$=INKEY$:IF e$="j" OR e$="n" THEN 1570 ELSE 1560
1570 IF e$="n" THEN PRINT#7,"Nochmal":GOTO 1390
1580 INPUT"Welche Nummer soll das Fenster haben ";i
1590 l(i)=l
1600 r(i)=r
1610 liu(i)=liu
1620 reu(i)=reu
1630 PRINT#7,"Das eben definierte Fenster hat nun"
1640 PRINT#7,"die WINDOW-Nr. ";i$;" ";i;" ";i$;" ."
1650 FOR t=1 TO 2000:NEXT
1660 MODE 1:CLS
1670 PRINT "Bitte wählen Sie: ":PRINT
1680 PRINT I$;" 1 ";i$;" = nächstes Fenster":PRINT
1690 PRINT I$;" 1 ";i$;" = Hauptmenü":PRINT
1700 e$=;" 2 ";i$;" OR e$="2" THEN 1717 ELSE 1700
1710 if E$="2" THEN RETURN ELSE GOTO 1390
1720 CLS
1730 MODE 1
1740 FOR i=1 TO 7
1750 IF 1(i)=0 THEN 1790
1760 WINDOW#i,l(i),r(i),liu(i),reu(i),:PAPER#i,2CLS#i
1770 e$"INKEY$:IF e$="" THEN 1770
1780 NEXT
1790 WINDOW#O,1,40,24,25: REM  Fenster für Nachricht
1800 PAPER#0,0:CLS#0:PEN#0,1
1810 PRINT#0,"Das war's! Bitte eine Taste drücken..."
1820 e$=INKEY$:IF e$="" THEN 1820
1830 RETURN
1840 CLS
1850 FOR i=1 TO 7
1860 PRINT" WINDOW#";i;",";l(i);",";r(i);",";
1870 PRINT liu(i);",";reu(i)
1880 PRINT
1890 NEXT
1900 LOCATE 1,19:PRINT"Bitte eine Taste drücken......"
1910 e$=INKEY$:IF e$="" THEN 1910
1920 RETURN
1930 CLS
1940 PRINT"Bitte machen Sie den Drucker startklar."
1950 PRINT"Wenn fertig, dann eine Taste drücken.."
1960 e$=INKEY§:IF e$="" THEN 1960
1970 FOR i=0 TO 7
1980 PRINT#8," WINDOW#";i;",";l(i);",";r(i);",";
1990 PRINT#8,liu(i);",";reu(i)
```

```
2000  PRINT
2010  NEXT
2020  LOCATE 1,19:PRINT"Bitte eine Taste drücken......"
2030  e$=INKEY§:IF e$="" THEN 1910
2040  RETURN
2050  BORDER 0
2060  MODE 1
2070  INK 1,0
2080  CLS
2090  FOR x=5 TO 35
2100  WINDOW #1,x,x,7,18
2110  PAPER#1,7
2120  CLS#1
2130  FOR p=1 TO 10:NEXT p
2140  NEXT
2150  WINDOW #1,5,35,7,18
2160  GOSUB 2310
2170  LOCATE 1,21
2180  BORDER 4,24
2190  SPEED INK 10,10
2200  FOR pause=1 TO 3000
2210  NEXT pause
2220  BORDER 0
2230  FOR i=35 TO 5 STEP -1
2240  WINDOW #2,i,i,7,18
2250  PAPER #2,1
2260  CLS#2
2270  FOR p=1 TO 300:NEXT p
2280  NEXT i
2290  n=n+1:IF n=2 THEN GOTO 2470
2300  GOTO 2050
2310  FOR a=1 TO 3:PRINT#1,"                 ":NEXT a
2320  text$="        SCHNEIDER CPC 464 "
2330  GOSUB 2420
2340  PRINT#1,"
2350  text$="        Beispiele für die"
2360  GOSUB 2420
2370  PRINT#1,"
2380  text$="        WINDOW-TECHNIK"
2390  GOSUB 2420
2400  PRINT#1
2410  RETURN
2420  FOR t=1 TO LEN(text$)
2430  PRINT#1, MID$(text$,t,1);
2440  FOR p=1 TO 10:NEXT p
2450  NEXT T
2460  RETURN
2470  INK 0,0:BORDER 0:MODE 1
2480  CLS
2490  PRINT
2500  PRINT
2510  PRINT"Window 1  Window 2   3    4"
2520  LOCATE 24,12
2530  PRINT" Window 5"
2540  LOCATE 24,12
2550  PRINT"Window 6   Window 7
2560  WINDOW#1,1,7,19,4
```

```
2570  PAPER#1,3
2580  CLS#1
2590  WINDOW#2,11,21,19,4
2600  PAPER#2,5
2610  CLS#2
2620  WINDOW#3,23,29,10,4
2630  PAPER#3,2
2640  CLS#3
2650  WINDOW#4,32,40,10,4
2660  PAPER#4,2
2670  CLS#4
2680  WINDOW#5,23,40,13,19
2690  PAPER#5,5
2700  CLS#5
2710  WINDOW#6,1,21,22,25
2720  PAPER#6,10
2730  CLS#6
2740  WINDOW#7,25,40,22,25
2750  FOR f=0 TO 15
2760  FOR i=1 TO 7:PAPER #i,f:CLS#i:NEXT
2770  FOR t=1 TO 1000:NEXT t
2780  NEXT f
2790  REM*********************
2800  FOR i=1 TO 200
2810  PRINT#1, "WINDOW"
2820  PRINT#2,i
2830  PRINT#3,i
2840  PRINT#4,i
2850  PRINT#5,i
2860  PRINT#6,i
2870  IF i= 100 THEN PRINT CHR$(7):PRINT#7,"ende"
2880  IF i= 200 THEN PRINT CHR$(7):CLS#7
2890  NEXT i
2900  MODE 0
2910  ON ERROR GOTO 3070
2920  FOR i=10 TO 100
2930  l=INT(RND(1)*(20-1)+1)
2940  r=INT(RND(1)*(40-21)+1)
2950  lu=INT(RND(1)*(24-1)+1)
2960  re=INT(RND(1)*(24-1)+1)
2970  farbe=RND(1)*15
2980  INK 1,farbe:PAPER#1,farbe
2990  WINDOW#1,l,r,lu,re:CLS#1
3000  t=RND(1)*400+10
3010  la=RND(1)*100+10
3020  SOUND 1,t,la
3030  SOUND 2,t+1,la
3040  FOR t=1 TO 100:NEXT
3050  NEXT
3060  FOR g=1 TO 2000:NEXT:RETURN
3070  GOTO 2900
```

Programmbeschreibung für den WINDOW-MAKER:

Zeile 1000 und 1010: Farben und Bildschirm einstellen Unterprogramm aufrufen (Menue zeichnen komplett)

Zeile 1020: Abfrage der Tastatur
Zeile 1030: Die Eingabe wird in einen numerischen Wert umgewandelt und geprüft. Dabei sind nur die Zahlen 1 bis 7 zulässig, mit Ausnahme der Zahl 6.
Zeile 1040 - 1090: Aufruf der einzelnen Unterprogramme entsprechend der gedrückten Zahl.
Zeile 1100: Rücksprung zum Menue
Zeile 1110 - 1380: Menue auf dem Bildschirm zeichnen und beschriften.

Das war nun der erste Baustein, der für die Auswahl der einzelnen Funktionen zuständig ist. Nachfolgend finden Sie jeden Programmpunkt gesondert aufgelistet und beschrieben.

Programmpunkt 1: WINDOW erstellen

Zeile 1390: Modus einstellen und Bildschirm löschen
Zeile 1400: WINDOW#7 für Meldungen definieren
Zeile 1410 - 1440: Abfrage der einzelnen Koordinaten für das neue Fenster
Zeile 1450: Modus einstellen und Bildschirm löschen
Zeile 1460: Textausgabe
Zeile 1470: Das eben erstellte Fenster zur Probe auf dem Bildschirm zeigen. Mit `PAPER#1,3` wird dieses Fenster mit der Farbe Blau gefüllt und mit `CLS#1` löscht man dieses Fenster komplett
Zeile 1480: Abfrage der Tastatur
Zeile 1490: Modus 1 anwählen
Zeile 1500 - 1550: Ausgabe der aktuellen Parameter für das eben erstellte Bildschirmfenster
Zeile 1560 - 1570: Tastaturabfrage `WINDOW` richtig oder falsch
Zeile 1580: Frage nach der Nummer des Fensters
Zeile 1590 - 1620: Übergabe der aktuellen Parameter in eine Tabelle
Zeile 1630 - 1640: Textausgabe in Bildschirmfenster mit der Nummer 7
Zeile 1650: Warteschleife
Zeile 1660: Bildschirmmodus wählen und Bildschirm löschen
Zeile 1670 - 1690: Menue für weitere Vorgehensweise
Zeile 1700: Abfrage der Tastatur
Zeile 1710: Wird die 2 gedrückt, dann Programm mit `RETURN` verlassen, sonst zur Zeile 1390 gehen.

Programmpunkt 2: WINDOW darstellen

Zeile 1720 -1730: Bildschirm löschen und Modus einstellen

Zeile 1740: Schleife zur Darstellung der erstellten Fenster öffnen (WINDOW#1 bis WINDOW#7).
Zeile 1750: Wenn die linke Position oben gleich Null ist, sind alle anderen Daten auch für dieses Fenster warscheinlich leer.
Zeile 1760: WINDOW darstellen, mit Farbe füllen und mit dieser Farbe das jeweilige Fenster zeichnen.
Zeile 1770: Abfrage, ob eine Taste gedrückt wurde
Zeile 1780: Schleife wieder schließen
Zeile 1790: Fenster für Nachrichten erstellen (WINDOW#0)
Zeile 1800: Farbe, Schrift und Löschen dieses Bildschirms
Zeile 1810: Textausgabe in Fenster Null
Zeile 1820: Abfrage der Tastatur
Zeile 1830: Rücksprung ins Menue mit RETURN

Programmpunkt 3: WINDOW-Parameter ausgeben

Zeile 1840: Bildschirm löschen
Zeile 1850 - 1890: Alle Parameter für das jeweilige Fenster ausgeben. Beispiel: WINDOW#2,6,8,3,9#
Zeile 1900: Textcursor positionieren und Text ausgeben
Zeile 1910: Abfrage der Tastatur, ob eine Taste gedrückt wurde
Zeile 1920: Rücksprung mit RETURN

Programmpunkt 4: WINDOW-Parameter ausdrucken

Zeile 1930 - 2040: Entspricht im Prinzip dem Punkt 3, nur mit dem Unterschied, daß hier mit PRINT#8 die Ausgaben auf den Drucker kommen.

Programmpunkt 5: WINDOW-DEMO zeigen

Zeile 2050: Rahmenfarbe auf Schwarz setzen
Zeile 2060: Bildschirmmodus 1 anwählen
Zeile 2070: In das Farbregister 1 die Farbnummer 0 setzen
Zeile 2080: Bildschirm löschen
Zeile 2090: Schleife zur Darstellung mehrerer WINDOWS öffnen
Zeile 2100: WINDOW mit den aktuellen Schleifenparametern aufbauen
Zeile 2110-2120: Farbe für WINDOW setzen und damit das Fenster füllen
Zeile 2130: Warteschleife abarbeiten
Zeile 2140: Schleife für WINDOW-Darstellung wieder schließen
Zeile 2150: WINDOW 1 aufbauen

Zeile 2160: Unterprogramm zur Beschriftung dieses Fensters aufrufen
Zeile 2170: Textcursor positionieren
Zeile 2180-2190: Bildschirmrahmen blinkt mit zwei verschiedenen Farben
Zeile 2200-2210: Warteschleifen durchlaufen
Zeile 2220: Nach Ablauf der Warteschleife Rahmen wieder auf Schwarz schalten
Zeile 2230-2280: WINDOW mit dem Text schrittweise wieder löschen
Zeile 2290: Zähler um eins erhöhen Ist der Zähler = 2 dann weiter mit dem nächsten WINDOW-DEMO
Zeile 2300: Sprung an den Anfang des Programms
Zeile 2310-2410: Unterprogramm zur Bezeichnung des WINDOWS
Zeile 2420-2460: Unterprogramm zur zeichenweisen Ausgabe von Texten
Zeile 2470-2480: Bildschirmfarbe komplett umschalten auf Schwarz, und den Bildschirm löschen
Zeile 2490-2500: Zwei Leerzeilen auf dem Bildschirm ausgeben
Zeile 2510-2730: Sechs WINDOWS erstellen und bezeichnen. Jedes Fenster erhält eine Farbe
Zeile 2740-2780: Mehrere Farbkombinationen durchspielen. Alle Fenster erhalten nacheinander eine bestimmte Farbe
Zeile 2790: Kommentarzeile
Zeile 2800: Schleife für 200 Durchläufe öffnen
Zeile 2810: In WINDOW 1 einen Text setzen
Zeile 2820-2860: In WINDOWS 2 bis 6 den aktuellen Wert der Schleife schreiben
Zeile 2870: Wenn die Schleife genau 100mal durchlaufen wurde, einen kurzen Piepton ausgeben und in Fenster 7 das Wort "ende" setzen
Zeile 2880: Wenn die Schleife 200 mal durchlaufen wurde, ebenfalls einen kurzen Ton ausgeben und Fenster 7 komplett löschen
Zeile 2890: Schleife mit NEXT wieder schließen
Zeile 2900: Bildschirm-Modus 0 wählen, da hier die meisten Farben möglich sind
Zeile 2910: Bei einem auftretenden Fehler in Zeile 3070 verzweigen
Zeile 2920: Schleife für 100 Durchläufe starten
Zeile 2930-2960: Zufallszahlen für die Lage des Fensters erzeugen
Zeile 2970: Zufällig eine Farbe wählen und diesen Wert in der Variablen Farbe speichern

Zeile 2980: In den Farbspeicher 1 die erzeugte Farbe bringen und aus diesem Farbspeicher die Farbe für das Fenster nehmen

Zeile 2990: WINDOW 1 anlegen und Bildschirm löschen. Damit hat jetzt das Fenster die Farbe aus Farbspeicher 1.

Zeile 3000: Zufälligen Wert für die Tonhöhe ermitteln

Zeile 3010: Zufälligen Wert für die Länge des Tons ermitteln

Zeile 3020: Mit dem Befehl SOUND den Kanal 1 ansprechen

Zeile 3030: Für den 2-Kanal-Effekt den Befehl SOUND etwas ändern. Dieses geschieht mit Hilfe der Tonhöhe

Zeile 3040: Warteschleife abarbeiten

Zeile 3050: Schleife für die Erzeugung von 100 WINDOWS wieder schließen

Zeile 3060: Warteschleife und Rücksprung ins Hauptmenue mit RETURN

Zeile 3070: Wenn ein Fehler auftrat, beginnt das Programm in Zeile 2900

17 Sound

Neben den ausgezeichneten Grafikmöglichkeiten verfügt der Schneider über einen speziellen Baustein zur Tonerzeugung, den 3-Kanal-Tongenerator-Baustein AY-3-8912. Unter Verwendung einiger BASIC-Befehle können damit die interessantesten Geräusche und Töne erzeugt werden und sogar das Abspielen ganzer Melodien wird möglich.

Zur Erzeugung von Tönen und Geräuschen stehen Ihnen folgende BASIC-Befehle zur Verfügung:

`SOUND`, `ENT` und `ENV`

Bei vielen Computern dieser Leistungsklasse fehlen diese Befehle, obwohl auch sie einen ähnlichen Baustein eingebaut haben. Bei anderen Geräten müssen meist recht umständlich mit dem Befehl POKE die erwünschten Parameter für diesen Baustein eingestellt werden. Das bedeutet zusätzliche Arbeit und - durch den POKE-Befehl bedingt - Verlust der Übersichtlichkeit. Sie sehen also, auch auf diesem Gebiet ist der Schneider anderen Computersystemen einen Schritt voraus.

Wir wollen nun gleich zu unserem ersten Beispiel übergehen. Viele Rechner haben einen einfachen Befehl, der einen kurzen Piepton erzeugt. Auch bei Ihrem CPC 464 gibt es einen solchen Befehl. Geben Sie einmal den Befehl

```
PRINT CHR$(7)
```

ein und drücken Sie die Taste <ENTER>. Jetzt sollte ein kurzer Piepton zu hören sein.

Sollten Sie nichts vernehmen, haben Sie vermutlich den Lautstärkeregler nicht genügend weit aufgedreht. Drehen Sie in diesem Fall den Regler, der an der rechten Seite des Tastaturgehäuses sitzt, noch ein wenig mehr auf.

Code Nr. 7 hat also die Aufgabe, einen kurzen Ton als einfaches Signal von sich zu geben. Dazu bedient man sich des Befehls

```
PRINT CHR$(7)
```

Damit haben wir die einfachste Möglichkeit der Tonerzeugung kennengelernt. Dieses Signal können Sie beispielsweise bei Fehlermeldungen oder wichtigen Hinweisen benutzen. Andere Geräusche sind mit diesem Befehl nicht zu erzeugen.

Der Befehl SOUND

Für die Erzeugung anderer Geräusche verwendet man den Befehl SOUND. Dieser Befehl ist sehr vielseitig, so daß es eine Reihe von Kombinationsmöglichkeiten gibt. Der Befehl SOUND kann wie folgt kombiniert werden:

```
SOUND Kanalnr., Frequenz, Tondauer,
Lautstärke, Hüllk. Lautstärke, Hüllk. für
Ton, Rauschcharakter
```

Es stehen Ihnen also insgesamt sieben verschiedene Parameter zur Verfügung. Seien Sie über diese Anzahl nicht beunruhigt, denn nicht alle Parameter werden gleichzeitig benutzt. Oft reichen schon wenige Angaben, um einen geeigneten Ton zu erzeugen.

Damit Sie aber auch alle Möglichkeiten nutzen können, möchte ich Ihnen als nächstes die einzelnen Parameter kurz vorstellen.

Die Kanalnr.:

Zur Tonerzeugung stehen Ihnen drei verschiedene Kanäle zur Verfügung, die Sie - je nach Anwendungsfall - miteinander kombinieren können. Diese Angabe ist zwingend notwendig, damit der Schneider weiß, auf welchem Kanal der Ton ausgegeben werden soll. Sie können wählen zwischen den Tonkanälen 1, 2 oder 4.

Die Frequenz:

Rein technisch gesehen ist es mit dem Schneider möglich, eine Maximalfrequenz von 62500 Hertz zu erreichen. Da dieser Bereich jedoch weit über dem für das menschliche Ohr noch hörbaren Bereich liegt, muß diese Frequenz immer niedriger als 62500 Hertz sein.

Ein Beispiel dazu. Ein normaler Mensch nimmt, je nach Alter, eine Frequenz von maximal ca. 16000 Hertz wahr. Tiere sind schon etwas empfindlicher. Für Hunde gibt es z.B. Pfeifen mit einer Frequenz über 16000 Hertz, die einen für den Menschen unhörbaren Ton erzeugen, der aber vom Hund noch wahrgenommen wird.

Was ist eine Frequenz? Vielleicht ist Ihnen der Begriff noch aus dem Physikunterricht geläufig. Zur Erinnerung noch einmal die physikalischen Zusammenhänge:

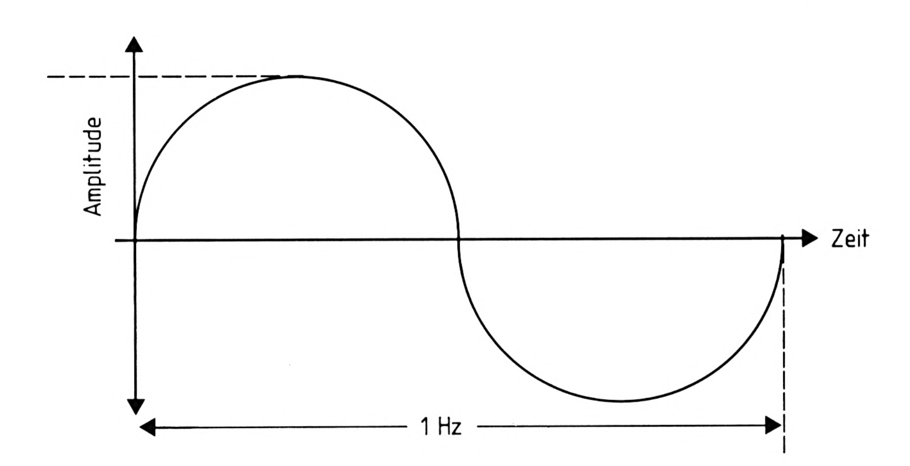


Bild 17.1: *Frequenzkurve*

Töne sind also nichts anderes als Schwingungen. Je mehr Schwingungen innerhalb einer Periode (t) entstehen, desto höher wird auch die Tonhöhe des Tons. Hier spricht man dann von der Frequenz. Je mehr Schwingungen also auftreten, umso höher ist die Frequenz.

Die Hüllkurve der Lautstärke

Die Hüllkurve dient dazu, einen Ton langsam an- oder abschwellen zu lassen. Damit können also Töne erzeugt werden, wie sie beispielsweise eine Gitarre oder eine Pauke liefert. Wenn man an einer Gitarrensaite zupft, entsteht ein Ton, der langsam abschwillt und leiser wird. Das folgende Beispiel verdeutlicht Ihnen diesen Zusammenhang:

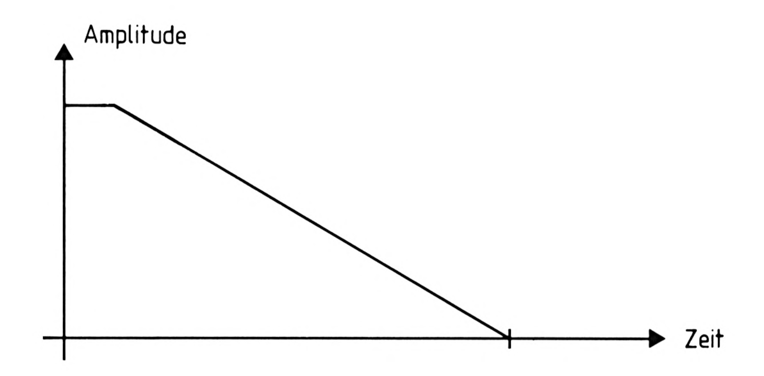


Bild 17.2: *Beispiel für die Lautstärkenhüllkurve einer Gitarrensaite*

Bei einem anderen Instrument ergibt sich vielleicht folgendes Bild:

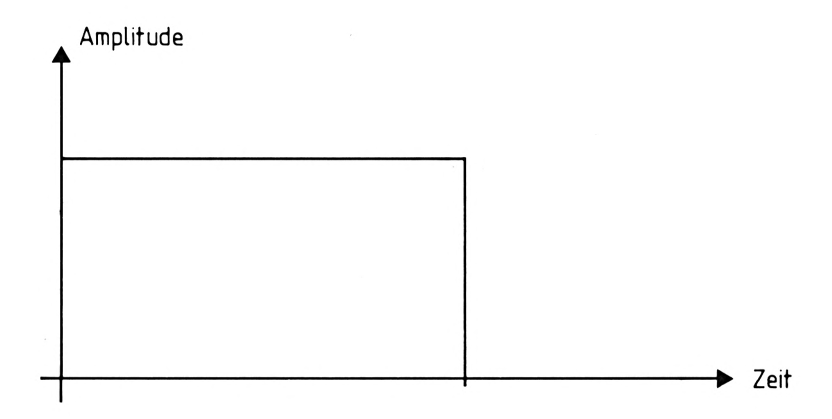


Bild 17.3: *Beispiel für die Lautstärkenhüllkurve für eine Mundflöte*

Die Frequenz wird mit der Einheit *Hertz* gemessen; dabei entspricht 1 Hz einer Schwingung pro Sekunde. Spricht man demgemäß von einem Ton

von 300 Hz, bedeutet das 300 Schwingungen pro Sekunde. die Frequenz errechnet sich aus folgender Formel:

```
FREQUENZ = 62500 / TEILER
```

Die Variable FREQUENZ wird später in den Befehl SOUND integriert. Der Wert 62500 ist die Grundfrequenz, die durch anschließendes Teilen mit der Variablen TEILER für den Menschen hörbar gemacht wird.

Die zulässigen Werte für die Variable TEILER liegen zwischen 1 und 4095. Damit sind Frequenzen möglich, die im Bereich von 15 bis 62500 Hz liegen.

Die Tondauer:

Hier wird angegeben, wie lange eine bestimmte Frequenz gehalten werden soll. Dabei wird mit der Zeiteinheit von 0.01 Sekunden gerechnet. Das bedeutet für uns, wenn wir einen Ton vier Sekunden lang hören möchten, geben wir an dieser Stelle den Wert 400 ein.

Zur Erklärung: Die vier Sekunden ergeben sich aus 400 mal 0.01 Zeiteinheiten. Möglich sind Zeitangaben bis zu 32767 Zeiteinheiten, was einer maximal möglichen Tondauer von etwas mehr als fünf Minuten entspricht.

Die Lautstärke:

Die Werte von 0 bis 15 sind hier als Angaben zulässig, wobei der Wert 15 für die größte Lautstärke steht. Die Lautstärke steht in engem Zusammenhang mit dem nächsten Parameter, der sogenannten Hüllkurve der Lautstärke.

Sie sehen, wie sich die einzelnen Musikinstrumente im Klangverhalten untereinander unterscheiden. Jedes Instrument besitzt also ein eigenes spezifisches Klanverhalten. Dieses spezielle Verhalten läßt sich auch mit dem Schneider CPC 464 realisieren, denn es stehen 15 verschiedene Möglichkeiten zur Verfügung. Damit haben Sie die Auswahl zwischen insgesamt 15 verscheidenen Klangverhalten, die Sie selbst programmieren können.

Welche der später programmierten Lautstärkehüllkurven verwendet wird, teilen Sie dem Schneider mit durch den Befehl

ENV

Die Hüllkurve für den Ton

Im Prinzip trifft das eben gesagte hier für den Ton zu, d.h. die Tonhöhe verändert sich am Anfang oder am Ende geringfügig. Ein Beispiel ist die Geige, deren Töne nicht immer genau gleich bleiben. Auch hier kann man wieder verschiedene Tonverhalten selbst programmieren. Dazu dient der Befehl

ENT

Auf die beiden Befehle ENV und ENT möchte ich nur am Rande eingehen, da sie für einen Programmierneuling recht kompliziert zu handhaben sind.

Der Rauschcharakter

Am Ende des Befehls SOUND steht ein Wert, der die Erzeugung von Rauschen unterstützt. Möglich sind dabei als Angaben die Werte 0 bis 31, wobei der höchste Wert ein sehr dumpfes und der niedrigste Wert ein sehr helles Rauschen hervorruft.

Sie kennen nun die einzelnen Parameter beim SOUND-Befehl, so daß wir uns als nächstes einige kleine Beispiele dazu ansehen wollen.

Aufgabe: Alternativ zum Befehl PRINT CHR$(7) soll mit Hilfe des Befehls SOUND ein anderer Ton erzeugt werden.

Verarbeitung:

SOUND 1, FREQUENZ

Ausgabe:

Kurzer Piepton in einer bestimmten Zeiteinheit

Beispiel:

Kanal-Nr. = 1, FREQUENZ=200
SOUND 1,200 eingeben und anschließend <ENTER> drücken.

Verwendung:

In allen Programmteilen, wenn ein akustisches Signal auf etwas hinweisen soll. Möglich ist beispielsweise, als Warnung eine sehr hohe Frequenz und für Nachrichten, die mit dem Ablauf des Programms nicht direkt etwas zu tun haben, eine niedrige Frequenz zu wählen.

Aufgabe: Ein Ton mit einer Dauer von drei Sekunden soll erzeugt werden.

Verarbeitung:

```
SOUND 1, FREQUENZ, TONDAUER
```

Ausgabe:

Ton mit einer Länge von drei Sekunden Dauer

Beispiel:

```
SOUND 1,500,300
```

Verwendung:

Unter anderem zum Abspielen von Melodien. Wird keine Zeitdauer angegeben, setzt der Schneider dafür automatisch seine Standardwerte ein. Der Standardwert für die Dauer beträgt rund eine Sekunde.

Aufgabe: Ein ansteigender Ton soll erzeugt werden.

Verarbeitung:

Mit Hilfe einer Schleife und dem Befehl SOUND soll dieser Ton entstehen.

Ausgaben:

Ansteigender Ton

Beispiel:

Kanal-Nr. = 2 FREQUENZ=variabel

Das Programm

```
10   FOR FREQUENZ = 900 TO 10 STEP -10
20   SOUND 2, FREQUENZ
30   NEXT FREQUENZ
```

Verbesserungen:

Die Zeit eines Tons soll verkürzt werden. Die Angabe der Tonlänge folgt als dritte Angabe nach dem Befehl SOUND. Wird nichts angegeben, setzt der Schneider einen Standardwert.

Ändern Sie einmal Zeile 20 wie folgt um:

```
20 SOUND 2, FREQUENZ, 3
```

Wenn Sie Ihr Programm nun erneut mit RUN starten, wird die Tonleiter wesentlich schneller abgespielt.

Verwendung:

Grundsätzlich in allen Videospielen gut einsetzbar.

Aufgabe: Es soll eine Sirene nachgeahmt werden. Dabei soll das Auf- und Abheulen mit Hilfe von zwei Tonkanälen realisiert werden.

Verarbeitung:

Eine Schleife zählt vorwärts und rückwärts. Die aktuellen Schleifenwerte werden zur Tonerzeugung im SOUND-Befehl verwendet. Diese Schleifenwerte werden zwei Tonkanälen zugewiesen, wobei der zweite Kanal frequenzmäßig etwas über dem ersten liegt.

Ausgabe:

Auf- und abschwellender Ton

Das Programm:

```
10    FOR FREQUENZ = 900 TO 100 STEP -10
20    SOUND 1, FREFQUENZ, 10
30    SOUND 2, FREQUENZ + 10, 10
40    NEXT FREQUENZ
50    FOR FREQUENZ = 100 TO 900 STEP 10
60    SOUND 1, FREQUENZ, 10
70    SOUND 2, FREQUENZ, 10
80    NEXT FREQUENZ
90    GOTO 10
```

Verbesserungen:

Auch hier gibt es wieder eine Reihe von Möglichkeiten. Als erstes können Sie die Zeitangabe, den letzten Wert im SOUND-Befehl, verändern. Dafür habe ich in meinem Beispielprogramm am Anfang eine Variable eingesetzt und in den SOUND-Befehlen die Angabe 10 gegen diese Variable ausgetauscht. Ebenfalls ändern könnte man Anfangs- und Endwerte oder die Schrittweite der FOR-NEXT-Schleifen.

Nachdem wir nun die Handhabung der Tonhöhe und Tondauer anhand einiger praktischer Beispiele kennengelernt haben, geht es in den folgenden Programmen um die Lautstärke. Sicherlich werden auch Sie überrascht sein, welche enorme Lautstärke der kleine in den Schneider eingebaute Lautsprecher hervorbringt!

Möglich sind bei der Angabe der Lautstärke die Werte von 0 bis 7. Arbeiten Sie darüber hinaus auch noch mit der Lautstärkenhüllkurve, stehen die Werte von 0 bis 15 zur Verfügung. Bei dem maximalen Wert von 15 ändert sich aber gegenüber der Version ohne Lautstärkenhüllkurve an der hörbaren Lautstärke nichts. Vielmehr wird der Bereich der möglichen Lautstärke bei der Programmierung von Lautstärkehüllkurven feiner aufgelöst.

Am besten ist es, wenn Sie als mögliche Parameter die Werte von 0 bis 7 verwenden, wobei der Wert 7 die größte und der Wert 0 die geringste bzw. gar keine Lautstärke erzeugt.

Aufgabe: Erzeugen eines lauten Tons

Verarbeitung:

`SOUND 1,100,200,LAUTSTÄRKE`

Ausgabe:

Lauter Ton

Beispiel:

`SOUND 1,100,200,7` ergibt einen lauten Ton

Aufgabe: Erzeugen eines leisen Tons

Verarbeitung:

Siehe oben

Ausgabe:

Leiser Ton

Beispiel: SOUND 1,100,200,1

Aufgabe: Der Ton eines Polizeiwagens soll erzeugt werden. Dabei soll mit der größtmöglichen Lautstärke gearbeitet werden.

Verarbeitung:

Einem Tonkanal zwei verschiedene Töne zuweisen und entsprechend lang den jeweiligen Ton halten. Der letzte Parameter soll die Lautstärke angeben.

Ausgabe:

Geräusch des Martinshorns

Beispiel

Verwendeter Tonkanal: 1
FREQUENZ 1: 180
FREQUENZ 2: 120
Zeiteinheiten: je 100
Lautstärke: 7

Das Programm

```
10    SOUND 1,180,100,7
20    SOUND 1,120,100,7
30    GOTO 10
```

Verwendung:

Für spezielle Effekte in Spielen und Anwendungen

Verbesserungen:

Selbstverständlich kann man recht einfach die Tonhöhe des Befehls SOUND verändern; aber auch die Länge des jeweiligen Tons ist variabel. Sehr schön hört es sich an, wenn Sie das Programm noch um die folgenden Zeilen erweitern:

```
15    SOUND 1,181,100,7
25    SOUND 1,121,100,7
```

18 Fehlermeldungen und Ihre Ursachen

18.1 Allgemeines zu Fehlermeldungen

Fehler treten beim Arbeiten mit Computern immer auf. Dies ist eine Tatsache, mit der man leben muß. Fehler sind also nie ganz auszuschließen. Fehler können zum einen hardwarebedingt oder zum anderen softwarebedingt sein. Ein Hardwarefehler liegt beispielsweise vor, wenn Sie Ihren Schneider einschalten und statt der gewohnten Meldung des Betriebssystems nur lauter undefinierbare Zeichen auf dem Bildschirm sehen. Ein anderer Hardwarefehler, der auftreten könnte: Ihre Diskette im Laufwerk ist defekt oder das Diskettenlaufwerk arbeitet nicht mehr einwandfrei.

Oft kommt es bei der Verwendung von Druckern vor, daß die Anschlußkabel nicht richtig fest sitzen und so durch Wackelkontakte Fehler auftreten können. Alle diese eben geschilderten Fehler sind reine Hardwarefehler, denen manchmal nur schwer auf die Schliche zu kommen ist. Besonders Hardwarefehler, die nur in unregelmäßigen Abständen auftreten, sind der Alptraum eines jeden Anwenders. Hier hilft meistens nur noch eine Fachwerkstatt, um dem Spiel ein Ende zu machen, denn schließlich möchte man ja mit einem einwandfreien Gerät arbeiten. Meine Erfahrung mit Hardwarefehlern ist folgende: Alles was nicht in den ersten vier Wochen Benutzung ausfällt, arbeitet oft noch Jahre später ohne große Probleme einwandfrei. Fehler treten meistens zuerst bei Verschleißteilen auf, wie z.B. bei der Tastatur, dem Diskettenlaufwerk und der Mechanik des Druckers. Bei häufiger Benutzung des Diskettenlaufwerks empfehle ich Ihnen, ungefähr alle sechs Monate den Schreib-Lesekopf neu justieren zu lassen. Es kann Ihnen sonst eventuell passieren, daß Sie einige Programme auf Diskette nicht mehr laden können.

Neben Verschleißerscheinungen ist Staub der größte Feind aller Computer. Eine regelmäßige Reinigung der Tastatur und eine Staubschutzhaube aus ganz normaler PVC-Folie ist da schon recht nützlich.

Solche Staubschutzhauben bekommt man schon relativ preisgünstig in den Computerabteilungen der Kaufhäuser und im Computerversandhandel. Soweit nun zu Fehlern, die bei der Hardware auftreten können.

Bei der Gruppe der Softwarefehler hat man es da schon wesentlich leichter. Hier kann nichts kaputt gehen oder etwas durchbrennen, wenn Sie einmal einen Befehl falsch eingegeben haben. Doch es gab mal eine Ausnahme, die mir persönlich bekannt ist. Bei der Verwendung eines bestimmten POKE-Befehls brannte bei einem Computermodell eines bekannten Herstellers ein Chip durch, wenn der Rechner über längere Zeit in Betrieb war. Dieser Fehler wurde aber bekannt und recht schnell behoben.

Das unangenehmste, was Ihnen passieren könnte bei einem Softwarefehler, ist das Aus- und Einschalten Ihres Schneiders. Zerstört wird also nichts dabei. Den Vorgang des Ausschaltens und erneuten Einschaltens sollten Sie zweckmäßigerweise mit den Tasten <SHIFT ESC CTRL> realisieren.

Die Gruppe der Softwarefehler möchte ich einmal in zwei Gruppen einteilen:

a. Die Schreibfehler beim Programmieren
b. Die logischen Fehler

Bei der Gruppe A haben Sie es noch am einfachsten, dem Fehler auf die Schliche zu kommen, denn hier haben Sie wahrscheinlich nur einen Befehl nicht richtig geschrieben. Hartnäckig können jedoch Fehler der Gruppe B sein, denn sie sind auf den ersten Blick nicht ohne weiteres zu erkennen.

Auf den nächsten Seiten finden Sie die Fehlermeldungen des Schneiders der Reihenfolge nach aufgeführt. Falls Sie einen Fehler in Ihren Programmen nicht finden, schlagen Sie dieses Kapitel auf und lesen sich den Text für diesen Fehler durch.

18.2 Alle Fehlermeldungen und typische Ursachen am praktischen Programmbeispiel

Ebenso wie man als Programmierer mit Variablen arbeitet, benutzt auch der Rechner bestimmte Variablen für seine Arbeit. Für Fehlermeldungen stehen zwei verschiedene Variablen zur Verfügung.

ERL bedeutet ERROR LINIE, also die Zeilennummer, in der ein Fehler auftrat

ERR bedeutet ERROR und gibt mit einer Zahl an, um welchen Fehler es sich dabei handelt.

Damit es bei auftretenden Softwarefehlern nicht zum Abbruch eines Programms kommt, kann man mögliche Fehler mit diesen Anweisungen lokalisieren und damit auch darauf reagieren. Beispielsweise könnte man dem Schneider mitteilen: Wenn die Fehlernummer 8 in der Zeile 110 auftritt, dann gehe zur Zeilennummer 1000 und mache dort mit dem Programmablauf weiter. Doch dazu später mehr. Nun die einzelnen Fehlermeldungen im Detail.

Fehlercode: 1

Bildschirmmeldung:

```
Unexpected NEXT in Zeilennummer ....
```

Ursache:

Die zur Schleife gehörenden FOR...TO Anweisung fehlt.

Beispiel:

```
10   CLS
20   PRINT I
30   NEXT I
```

Hier fehlt also eindeutig die FOR...TO Anweisung zwischen der Zeile 10 und 20.

Behebung des Fehlers:

Einfach die fehlende Zeile eintragen.

Fehlercode: 2

Bildschirmmeldung:

```
Syntax error in Zeilennummer .......
```

oder im Direktmodus einfach

```
Syntax error
```

Ursache:

Ein BASIC-Befehl wurde nicht richtig geschrieben oder die BASIC-Befehle haben keinen Leerraum untereinander oder es fehlen irgendwelche Zeichen zu einer bestimmten Anweisung.

Beispiel:

```
10    MODE 1
20    CLS
30    PRIND"Ende"
40    GOTO 10
```

Dieses Programm unterbricht mit der folgenden Fehlermeldung: `Syntax error in 30`. Sieht man sich die Zeile 30 genau an, erkennt man dort, daß der Befehl PRINT mit einem "D" geschrieben wurde. Einen solchen Befehl gibt es aber nicht und darum kam es zu dieser Fehlermeldung in Zeile 30.

Behebung des Fehlers:

Entweder die Zeile komplett neu schreiben, was ich persönlich bei kurzen Zeilen bevorzuge oder mit dem Copy-Cursor die Zeile editieren und berichtigen.

Bei einem solchen Fehler im Direktmodus einfach den entsprechenden Befehl noch einmal eingeben.

Anmerkung:

Dieser Fehler ist der häufigste Fehler überhaupt und läßt sich ohne große Schwierigkeiten finden. Es handelt sich also bei diesem Fehler meistens um einen reinen Schreibfehler.

Fehlercode:3

Bildschirmmeldung:

`Unexpected RETURN in Zeilennummer .....`

Ursache:

Der Befehl RETURN wurde gefunden, aber es fehlt der dazugehörige Befehl GOSUB.

Beispiel:

```
10    PRINT"Unterprogramm aufrufen"
20    GOSUB 1000
30    PRINT"Ende"
40    END
1000  CLS
```

```
1010  PRINT"Unterprogramm beginnt hier"
1020  RETURN
```

Wird das Programm normal mit RUN gestartet, arbeitet es einwandfrei alle Zeilennummern ab.

Beim Testen von Unterprogrammen wird man aber meistens direkt in das Programm springen und hier in unserem Beispiel vielleicht zur Zeile 1000 gehen. Man könnte dies mit GOTO 1000 oder RUN 1000 erreichen. Das Unterprogramm liefe zwar einwandfrei, doch wegen dem RETURN in der letzten Zeilennummer tritt dann der Fehler auf:

```
Unexpected RETURN in 1020
```

In allen anderen Fällen, in denen dieser Fehler auftritt, haben Sie einfach den Befehl GOSUB irgendwo im Programm vergessen.

Behebung des Fehlers:

In unserem ersten Fall oben brauchen Sie nichts zu tun, denn der GOSUB-Befehl ist ja vorhanden. In den anderen Fällen müßten Sie sich die Stelle suchen, an der der Befehl GOSUB normalerweise hingehört und ihn einfach einfügen als zusätzliche Programmzeile.

Anmerkung:

Die häufigste Fehlerursache möchte ich Ihnen hier auch kurz vorstellen. Diesem Fehler ist im Prinzip nicht so einfach auf die Schliche zu kommen, denn der Befehl GOSUB ist in diesem Fall nämlich vorhanden!

Gehen wir einmal von folgender Überlegung aus: Wir haben uns ein Programm aufgebaut, das aus einem Hauptteil und mehreren Unterprogrammen am Ende besteht. Dies könnte beispielsweise so aussehen:

```
100   REM hier beginnt nun das Hauptprogramm
200   GOSUB 1000
....
....
500   GOSUB 2000
600   GOSUB 3000
....
....
900   REM hier ist das Programm irgendwo zuende
1000  REM hier beginnt das erste Unterprogramm
....
....
1900  RETURN
```

```
2000  REM hier fängt das zweite Unterprogramm an
....
....
2900  RETURN
3000  REM hier nun das dritte Unterprogramm
3900  RETURN
```

Wird ein solches Programm gestartet mit RUN, arbeitet es alle Unterprogramme einwandfrei ab, denn die Befehle GOSUB sind ja vorhanden. Trotzdem kommt es später in 1900 zur Fehlermeldung `Unexpected RETURN in 1900`. Die Lösung ist ganz einfach: Bei dem Hauptprogramm fehlt der Befehl END nach der Zeile 900, deshalb wird das Unterprogramm zum 2.ten Mal durchlaufen!

Fehlercode: 4

Bildschirmmeldung:

```
DATA exhausted in Zeilennummer.....
```

Ursache:

Es sind nicht genügend Daten zum Lesen für die READ-Anweisung vorhanden.

Beispiel:

```
10    MODE 1
20    CLS
30    DATA eins,zwei,drei,vier,fünf,sechs,sieben
40    FOR LESEN=1 TO 8
50    READ BEGRIFF$
60    PRINT BEGRIFF$
70    NEXT LESEN
```

Starten Sie dieses Programm mit RUN, dann erscheint zum Schluß die Fehlermeldung:

```
DATA exhausted in 50
```

Das Programm liest nacheinander die einzelnen Texte und gibt sie auf dem Bildschirm aus. In der Zeile 40 teilt man dem Rechner mit, daß er 8 mal einen Begriff lesen soll, aber es sind in der DATA-Anweisung nur 7 Begriffe vorhanden.

Eine weitere Ursache ist das erneute Lesen von Daten, die vorher schon irgendwo mit READ eingelesen wurden und nun nochmals gelesen werden sollen. Hier hilft nur der Befehl RESTORE, der den

DATA-Zeiger wieder auf die erste Angabe der ersten DATA-Zeile setzt.

Behebung des Fehlers:

Entweder dafür sorgen, daß genügend Daten zum Lesen vorhanden sind oder die Anweisung RESTORE einsetzen, die den DATA-Zeiger wieder auf den Anfang des Datenbestands setzt.

Fehlercode: 5

Bildschirmmeldung:

```
Improper argument in Zeilennummer ....
```

oder im Direktmodus nur

```
Improper argument
```

Ursache:

Bei Befehlen, in denen Zahlen irgendwie in Klammern oder nach dem Befehl stehen, stimmt etwas nicht. Meistens sind diese Werte zu groß angegeben.

Beispiel:

```
10    MODE 1
20    CLS
30    FOR I=32 TO 257
40    PRINT CHR$(I)
50    NEXT I
```

Dieses Programm zeigt Ihnen zuerst alle möglichen Zeichen Ihres Schneiders und bricht dann mit folgender Fehlermeldung ab:

```
Improper argument in 40
```

Der Grund: Sie haben versucht, mehr als 255 Zeichen zu lesen. Es sind nämlich nur soviel zum Arbeiten vorhanden. Da der Endwert in der Schleife aber 257 lautet, trat dieser Fehler auf.
Ein weiteres Beispiel im Direktmodus eingegeben:
`PRINT SPACE$(1000)` erzeugt ebenfalls diese Meldung, jedoch ohne Angabe der Zeilennummer. Eine Fehlermeldung kommt hier, weil nur 255 als Angabe in Klammern zulässig ist.
Versuchen Sie doch einmal, mit MODE 3 einen neuen Bildschirmmodus aufzurufen. Die werden sehen, auch dies erzeugt die oben angegebene Fehlermeldung.

Behebung des Fehlers:

Den entsprechenden Befehl neu eingeben oder falls er in einer Zeile steht, dort korrigieren.

Fehlercode: 6

Bildschirmmeldung:

`Overflow in Zeilennummer .....`

Ursache:

Zahlenüberlauf; Berechnungen liegen außerhalb des Bereichs, den BASIC verarbeiten kann.

Beispiel:

```
10    MODE 1
20    CLS
30    FOR I=0 TO 200
40    PRINT 2^I
50    A$=INKEY$: IF A$="" THEN 50
60    NEXT I
```

Starten Sie dieses kleine Programm und sehen Sie sich in aller Ruhe an, wie die Werte immer größer werden. Irgendwann wird dann die Zahl so groß, daß sie von BASIC aus nicht mehr verarbeitet werden kann. Besonderheit dabei: Das Programm wird nicht abgebrochen!

Behebung des Fehlers:

Nach Möglichkeit Werte vermeiden, die mit solchen riesigen Größen arbeiten.

Fehlercode: 7

Bildschirmmeldung:

`Memory full in Zeilennummer .......`

Ursache:

Arbeitsspeicher vollständig belegt

Beispiel:

```
10    N=1000
20    N=N+100
30    DIM A(N)
40    LOCATE 10,10
```

```
50    PRINT FRE(1)
60    ERASE A
70    GOTO 20
```

Dieses Programm dimensioniert in Schritten von 100 die Variable A und gibt anschließend den noch freien Speicherplatz mit FRE(1) aus. Dann wird die Variable A wieder mit einer höheren Zahl dimensioniert.

Das Programm wird irgendwann mit der Fehlermeldung Memory full in 30 abbrechen. In dieser Zeile wird dimensioniert und zwar beim letzten Mal so groß, daß der Rechner dies nicht mehr verarbeiten kann.

Aber auch andere Möglichkeiten können bei diesem Fehler vorliegen. Beispielsweise tritt dieser Fehler bei sehr vielen verschachtelten Schleifen und Unterprogrammen auf. Weitere Ursachen können sein:
- das BASIC-Listing ist zu groß
- es sind zu viele Variablen verwendet worden
- der Speicherplatz für BASIC wurde mit dem Befehl MEMORY zu gering dimensioniert

Behebung des Fehlers:

Als erstes würde ich mir die Zeilennummer auflisten, in der dieser Fehler auftrat. Meistens handelt es sich dabei um eine DIM-Anweisung, die nicht mehr ausgeführt werden kann, weil einfach kein Speicherplatz mehr zur Verfügung steht.

Setzen Sie den DIM-Wert niedriger an oder schaffen Sie mit dem Befehl ERASE Platz für neue Variablen. Nutzen Sie beim Programmieren die Funktion FRE(1), die Ihnen bekanntlich jederzeit angibt, was einem an Arbeitsspeicher noch zur Verfügung steht.

Bei langen BASIC-Programmen: Arbeiten Sie wo es irgendwie geht mit Unterprogrammen. Überprüfen Sie von Zeit zu Zeit Ihren eigenen Programmierstil. Definieren Sie alle Variablen mit einfacher Genauigkeit, denn auch so kann man schon eine ganze Menge an Speicherplatz sparen. Falls Ihr BASIC-Programm später einwandfrei läuft, löschen Sie alle REM-Befehle aus dem Programm.

Doch Vorsicht!! Achten Sie darauf, daß es sich bei den Zeilennummern, die Sie jetzt löschen wollen, nicht um Stellen handelt, auf die Sie später mit dem GOTO-Befehl springen wollen.

Fehlercode: 8

Bildschirmmeldung:

```
Line does not exist in Zeilennummer......
```

Ursache:

Eine Zeilennummer, die angesprungen werden soll, existiert im Programm nicht.

Beispiel:

```
10   MODE
20   CLS
30   GOTO 60
40   PRINT"Programmende"
50   END
70   GOTO 40
```

Nach Eingabe von RUN startet sich dieses kleine Programm und gibt folgende Fehlermeldung aus:

```
Line does not exist in 30
```

In der Zeile 30 steht also eine Zeilennummer, die das Programm nicht finden kann, weil sie überhaupt nicht existiert. Dieser Fehler tritt recht häufig in Zusammenhang mit den Befehlen GOTO und GOSUB auf. Aber auch Programme, in denen Teile mit dem Befehl RENUM umnumeriert werden, sind oft Fehlerquellen.

Behebung des Fehlers:

Listen Sie sich den Bereich der angegebenen Zeilennummer auf. In unserem Fall die Zeilennummern von 40 bis 70. Sie werden dann sehr schnell sehen, wo die Zeilennummer fehlt. Geben Sie anschließend die gewünschte Zeilennummer und den Befehl ein. Wenn Sie nun Ihr Programm erneut starten, müßte dieser Fehler eliminiert sein.

Fehlercode: 9

Bildschirmmeldung:

```
Subscript out of range in Zeilennummer...
```

Ursache:

Es wurde versucht, einen Bereich anzusprechen, der außerhalb des dimensionierten Bereichs lag. Dieser Fehler tritt meistens im Zusammenhang mit Schleifen und indizierten Variablen auf.

Beispiel:

```
10    DIM A(40)
20    FOR I=1 TO 45
30    PRINT SQR(I)
40    A(I)=SQR(I)
50    NEXT I
```

Das hier vorgestellte Programm wird mit der Fehlermeldung `Subscript out of range in 40` den Ablauf unterbrechen.

Behebung des Fehlers:

Dieser Fehler ist relativ leicht zu finden, denn es gibt eigentlich nur zwei Punkte, wo man suchen muß.

Der erste Punkt:
Kontrollieren Sie die `DIM`-Anweisung. Ist sie zu klein geraten, dann setzen Sie sie einfach höher.

Der zweite Punkt:
Ist der Endwert der Schleife wirklich so hoch oder kann man ihn herabsetzen?

Auch darauf sollten Sie achten.

Fehlercode: 10

Bildschirmmeldung:

`Array already dimensioned in Zeilennummer ..`

Ursache:

Es wurde versucht, eine indizierte Variable erneut zu dimensionieren.

Beispiel:

```
10    MODE 1
20    CLS
30    DIM A(100), B(100), C(100)
40    FOR I=1 TO 100
50    A(I)=SIN(I)
60    B(I)=I*2
70    C(I)=B(I)-A(I)
```

```
80    PRINT A(I), B(I), C(I)
90    NEXT I
100   PRINT"Nochmal?  J=JA  N=NEIN"
110   E$=INKEY$: IF E$="" THEN 110
120   IF E$="J" THEN GOTO 10
130   END
```

Kurz zur Funktion des Programms:

In der Zeile 30 werden 3 verschiedene Felder dimensioniert, die in den Zeilen 50 bis 80 für Berechnungen benötigt werden. Nachdem alle Werte berechnet und ausgedruckt worden sind, kommt zum Schluß die Frage:

```
Nochmal? J=JA   N=NEIN
```

Geben Sie den Buchstaben J ein, beginnt das Programm wieder einwandfrei in der Zeile 10, bringt dann aber in Zeilennummer 30 die Meldung: `Array already dimensioned in 30`. Es wurde also versucht, ein bereits dimensioniertes Feld erneut zu reservieren.

Behebung des Fehlers:

Hier gibt es wieder mehrere Möglichkeiten, die sich von Fall zu Fall anwenden lassen.

Möglichkeit 1:

Sie löschen mit dem Befehl **ERASE** die verwendeten Variablenfelder und können diese dann erneut dimensionieren.

Möglichkeit 2:

Wenn Sie den Bereich nicht löschen möchten, dann wählen Sie einfach eine andere Variablenbezeichnung für die nächsten Variablen. Oft ist dies recht nützlich, senkt aber den freien Arbeitsspeicher erheblich.

Möglichkeit 3:

Sie gewöhnen sich einen sauberen Programmstil an und legen solche Programmteile, wo dimensioniert wird, an den Anfang eines Programms.

Später sorgen Sie dafür, daß dieser Programmteil nur ein einziges Mal durchlaufen wird und dann nicht mehr. Falls dann noch immer Fehler auftreten sollten, wissen Sie ganz genau, an welcher Stelle Sie nachsehen müssen.

Fehlercode: 11

Bildschirmmeldung:

```
Division by zero
```

Ursache:

Sie haben einen Wert durch Null dividiert. Nach den mathematischen Regeln ist dies nicht zulässig, deshalb erscheint dieser Fehler.

Beispiel:

```
10    CLS
20    PRINT 200/A
```

Als Ergebnis werden Sie hier erhalten:

```
Division by zero
```

Es wurde also versucht, den Wert 200 durch A zu teilen, obwohl der Wert für A nirgendwo definiert wurde und deshalb Null ist.

Besonderheit dieser Fehlermeldung: Das Programm wird dabei nicht unterbrochen, sondern setzt seine Ausführung fort.

Bitte beachten Sie, daß dies unter bestimmten Umständen auch zu Folgefehlern in den weiteren Berechnungen kommen kann, die man auf den ersten Blick nicht erkennt.

Behebung des Fehlers:

Unterbrechen Sie Ihr Programm und fügen Sie die folgenden Zeilennummern zusätzlich ein:

```
1     ON ERROR GOTO 60000
60000 MODE 1:CLS
60010 PRINT"Fehler in Zeile ";ERL
60020 IF ERR=11 THEN PRINT"Sie haben hier durch den Wert Null geteilt!": END
```

Diese Fehlerbehandlungsroutine für diesen Fehler hilft Ihnen bei der Suche. Sie brauchen nur noch die angegebene Zeilennummer listen.

Fehlercode: 12

Bildschirmmeldung:

```
Invalid direct command
```

Ursache:

Sie haben einen BASIC-Befehl eingegeben, der nur innerhalb eines Programms benutzt werden darf.

Beispiel:

Beim Definieren einer Benutzerfunktion im Direktmodus tritt diese Meldung beispielsweise auf.

Behebung des Fehlers:

Diesen Befehl nur innerhalb eines Programms verwenden. Dieser Fehler tritt sehr selten auf.

Fehlercode: 13

Bildschirmmeldung:

```
Type mismatch in Zeilennummer.....
```

Ursache:

Sie haben einen falschen Variablentyp erwischt. Meistens handelt es sich hier um einen Schreibfehler beim Programmieren, wo man nur das Dollarzeichen vergessen hat.

Beispiel:

```
10     INPUT"Bitte wählen: 1 oder 2  ",A$
20     IF A$=1 OR A$=2 THEN GOTO 100
30     PRINT"Falsch.  Nochmal"
40     GOTO 10
100    REM  hier weiter im Programm
```

Behebung des Fehlers:

Listen Sie die in der Fehlermeldung angegebene Zeilennummer und korrigieren den Fehler. In unserem Beispiel ist die Zeile 20 falsch, denn wenn man schon alphanumerische Daten prüft, dann müssen diese auch in Anführungsstrichen stehen.

Die Zeile 20 müßte also richtig lauten:

```
20 IF A$="1" OR A$="2" THEN GOTO 100
```

Fehlercode: 14

Bildschirmmeldung:

```
String space full in Zeilennummer...
```

Ursache:

Es wurde zuviel Speicherplatz für Variablen verwendet und zwar für die alphanumerischen Variablen.

Beispiel:

```
10    A$=STRING$(255,45)
20    AB$=STRING$(255,45)
......
......
......
5000  ZZ$=STRING$(255,45)
```

Ich habe dieses Beispiel einmal ausprobiert und bin erst nach vielen eingegebenen Variablen zu dieser Fehlermeldung gekommen. Bei meiner Arbeit mit dem Schneider ist bei mir dieser Fehler noch nicht aufgetreten. In umfangreichen Programmen könnte dieser Fehler aber auftreten, wenn nicht mehr verwendete Variablen im Programm bleiben und man vergessen hat, diese aus Speicherplatzersparnis zu löschen.

Behebung des Fehlers:

Nicht mehr benötigte Strings mit ERASE löschen.

Fehlercode: 15

Bildschirmmeldung:

```
String too long in Zeilennummer.....
```

Ursache:

Eine verwendete Zeichenkette enthält mehr als 255 Zeichen. Nur maximal 255 Zeichen lassen sich unterbringen.

Beispiel:

```
10    TEXT$=""
20    TEXT$=TEXT$+"#"
30    PRINT TEXT$
40    GOTO 20
```

Dieses kleine Programm läßt die Variable TEXT$ immer größer werden. Irgendwann ist dann die maximale Stringgröße erreicht und der Rechner gibt die Meldung aus:

```
String too long in 20
```

Wie Sie eben gesehen haben, lassen sich nur 255 Zeichen in einer Variablen unterbringen.

Behebung des Fehlers:

Meistens tritt dieser Fehler bei der Addition von Texten auf, auch wenn dies auf den ersten Blick nicht so ersichtlich ist. Benutzen Sie eine neue Stringvariable oder setzen die alte mit z.B. `TEXT$=""` wieder auf den Anfangswert zurück.

Noch ein Hinweis: Je mehr Variablen man in einem Programm verwendet, desto unübersichtlicher wird es und Fehler schleichen sich leichter ein. Wenn Sie sich daran orientieren, kann eigentlich nicht mehr viel schief gehen.

Fehlercode: 16

Bildschirmmeldung:

```
String expression too complex in Zeilennummer
```

Ursache:

Eine Verarbeitung von Zeichenketten ist für das System nicht möglich, da zuviele Anweisungen miteinander verschachtelt wurden.

Beispiel:

Ich habe bisher diese Fehlermeldung noch nie mit einer Zeilennummer erzeugt. Falls der Rechner einmal auf diesen Fehler stoßen sollte, hat der Programmierer mit Sicherheit schon den Überblick über die Funktionsweise dieser Zeilennummer verloren.

Behebung des Fehlers:

Teilen Sie den gesamten Inhalt der Zeilennummer in einzelne Teilschritte ein. Dies dient auch Ihnen zur besseren Erkennung von Fehlern.

Fehlercode: 17

Bildschirmmeldung:

```
Cannot CONTinue
```

Ursache:

Ein Programm wurde entweder mit dem Befehl STOP oder mit der Taste <ESC> abgebrochen. Sie haben dann etwas im Programm geändert oder vielleicht eine Zeilennummer zusätzlich eingefügt. Wenn anschließend nun versucht wird, das Programm mit CONT fortzusetzen, erscheint diese Fehlermeldung.

Beispiel:

```
10   N=0
20   N=N+10
30   PRINT N
40   STOP
50   GOTO 20
```

Sie lassen das Programm bis zur Zeile 40 laufen, wo es dann automatisch durch den STOP-Befehl unterbrochen wird.
Fügen Sie jetzt z.B. die Zeile 5 ein und schreiben:

```
5 REM --- Testprogramm ----
```

dann kann mit CONT das Programm schon nicht mehr in der Zeile 40 veranlaßt werden, mit dem Ablauf weiter zu machen.

Behebung des Fehlers:

Beim Auftreten dieses Fehlers nichts mehr am Programm ändern. Wenn Sie trotzdem etwas ändern, starten Sie am besten danach wieder mit dem Befehl RUN Ihr BASIC-Programm.

Fehlercode: 18

Bildschirmmeldung:

```
Unknown user function in Zeilennummer
```

Ursache:

Eine Funktion, die aufgerufen wurde, war noch nicht mit dem Befehl DEF FN definiert.

Beispiel:

Sie haben ein Programm geschrieben, in dem immer wieder komplizierte Berechnungen verarbeitet werden sollen. Diese Berechnungen haben Sie am Anfang des Programms mit der Anweisung DEF FN definiert und rufen sie innerhalb des Programms immer wieder mit der Funktion FN auf. Fehlt am Anfang diese Definition, dann erscheint diese Fehlermeldung.

Behebung des Fehlers:

Am Anfang des Programms diese Funktion definieren.

Fehlercode: 19

Bildschirmmeldung:

```
RESUME missing in Zeilennummer.....
```

Ursache:

Mit der Anweisung ON ERROR GOTO wurde am Anfang des Programms festgelegt, daß in eine Fehlerbehandlungsroutine gesprungen werden sollte. Ist das Programm dann irgendwann in dieser Routine, dann braucht es später auch noch einen Hinweis, was geschehen soll, wenn der Fehler lokalisiert worden ist.

Normalerweise setzt man hier den Befehl RESUME ein, der das Programm wieder an der Stelle weitermachen läßt, wo der Fehler auftrat. Fehlt in einer solchen Behandlungsroutine der Befehl RESUME, dann kommt es zwangsläufig zu dieser Fehlermeldung.

Beispiel:

```
10    ON ERROR GOTO 1000
20    PRENT"Diese Zeile enthält einen Fehler!"
30    REM
1000  MODE 1:CLS
1010  PRINT"Fehler in Zeile: ";ERL
1020  PRINT"Fehlercode:      ",ERR
```

Dieses Programm verzweigt zur Zeile 1000, weil in der Zeile 20 der Befehl PRINT nicht richtig geschrieben wurde.

Jetzt erscheint die zweite Fehlermeldung und zwar:

```
RESUME missing in 1020
```

Behebung des Fehlers:

Wenn das Programm nicht mehr weitermachen soll, dann setzen Sie in 1030 den Befehl END. Sonst reicht der Befehl RESUME in Zeile 1030 voll aus.

Fehlercode: 20

Bildschirmmeldung:

`Unexpected RESUME in Zeilennummer....`

Ursache:

In einem Programm wurde der Befehl RESUME gefunden, der nicht im Zusammenhang mit einer Fehlerbehandlungsroutine steht. Dieser Befehl kann aber nur im Zusammenhang mit einer solchen Routine verwendet werden. Jede andere Verwendung führt zu dieser Fehlermeldung.

Beispiel:

```
10    ON ERROR GOTO 1000
20    PRINT"DEMOPROGRAMM"
30    PRINT
40    PRINT"Nun kommt der Fehler in Zeile 50"
50    RESUME
60    END
1000  MODE 1:CLS
1010  PRINT"Fehlerzeile:    ";ERL
1020  PRINT"Fehlercode:     ";ERR
1030  RESUME
```

Geben Sie als erstes die Zeilen 20 bis 60 ein und starten das Programm mit RUN. Sie erhalten als Reaktion auf diesen Fehler die Meldung:

`Unexpected RESUME in 50`

Geben Sie nun die folgenden Zeilennummern ein. Auch dieser Fehler wird erkannt und in den Zeilen 1010 und 1020 lokalisiert.

Behebung des Fehlers:

Die Zeile mit dem Befehl RESUME löschen, wenn es sich nicht um einen versehentlichen Sprung in eine solche Fehlerbehandlungsroutine handelt.

Fehlercode: 21

Bildschirmmeldung:

`Direct command found`

Ursache:

Dieser Fehler kann beim Laden eines Programms oder einer Datei von Kassette vorkommen.

Beispiel:

Sie haben Ihr Programm ordnungsgemäß gespeichert. Jetzt könnten aber bei der Datenspeicherung Fehler aufgetreten sein oder beim Lesen gibt es Schwierigkeiten.

Meistens handelt es sich aber nicht um Programme, sondern um Dateien, bei denen dieser Fehler auftritt.

Behebung des Fehlers:

Versuchen Sie den gesamten Ladevorgang zu wiederholen, mit ein wenig Glück klappt es vielleicht beim nächsten Anlauf. Fehler, die beim Laden auftreten, sind immer sehr ärgerlich, denn für die Daten hat man ja schon viel Arbeit hineingesteckt. Lesen Sie sich vielleicht noch einmal kurz das Kapitel durch, in dem das Laden und Speichern von Programmen besprochen wird.

Fehlercode: 22

Bildschirmmeldung:

```
Operand missing in Zeilennummer....
```

Ursache:

In der angegebenen Zeilennummer bei der Fehlermeldung fehlt ein Teil einer Berechnung oder eine zusätzliche Angabe bei einem BASIC-Befehl.

Beispiel:

```
10    CLS
20    PRINT"Es folgt nun der erste Fehler...."
30    PRINT 12+144-
40    PRINT
50    PRINT"Es kommt der nächste Fehler...."
60    MODE
70    PRINT"Das war's..."
```

Sie haben sicherlich schon bemerkt, daß in der Zeilennummer 30 noch eine Angabe für die Berechnung fehlt.

Das war ein kleines Beispiel für fehlende Werte innerhalb von Berechnungen. Ändern Sie die Zeile 30 und geben dahinter irgend-

eine Zahl ein, damit diese Fehlermeldung nicht mehr auftritt. Starten Sie nun das ganze Programm nochmal mit RUN. In Zeile 60 bleibt das Programm mit der gleichen Fehlermeldung stecken. Hier fehlt die Angabe bei einem BASIC-Befehl. Mit `60 MODE 1` wäre die Sache im Prinzip schon gelaufen, denn nun hat der Befehl ja seine nötige Angabe.

Behebung des Fehlers:

Zeile auflisten und entsprechend ändern.

Fehlercode: 23

Bildschirmmeldung:

```
Line too long in Zeilennummer....
```

Ursache:

Eine Zeilennummer ist zu lang.

Beispiel:

Diese Fehlermeldung können Sie schon fast ausschließen, den beim Programmieren meldet sich der Schneider schon automatisch mit einem Piepton, wenn die Programmierzeile bis hinten voll ist. Bekanntlich kann eine Zeile insgesamt 255 Zeichen verarbeiten.

Behebung des Fehlers:

Sollte aus irgendeinem Grund dieser Fehler bei Ihnen auftauchen, schreiben Sie ganz einfach die letzten Befehle der Zeilennummer in eine neue Zeilennummer. Damit haben Sie den Fehler schon beseitigt.

Fehlercode: 24

Bildschirmmeldung:

```
EOF met
```

Ursache:

Das Dateiende wurde beim Laden einer Datei bereits erreicht, aber trotzdem lädt der Recoder weiter.

Beispiel:

Arbeiten Sie hier am besten mit der Abfrage EOF, dem END OF FILE. Ist das Dateiende erreicht, dann wird der Ladevorgang beendet.

Behebung des Fehlers:

Benutzung des Befehls EOF

Beim Arbeiten mit dem Diskettenlaufwerk sollte ich einmal mehrere Unterprogramme mit dem Befehl MERGE zusammenbinden. Beim Cassettenlaufwerk war das überhaupt kein Problem, umso erstaunter war ich, als auf dem Bildschirm die Meldung EOF MET erschien. Sie haben die gleichen Probleme, wie ich sie beim MERGEn hatte? Dann sollten Sie sich die nächsten Zeilen einmal ansehen:

Programme auf Diskette werden bekanntlich nicht so abgespeichert wie man sie auf dem Bildschirm in Form eines Listings sieht. Vielmehr werden die einzelnen Befehle in eine für den Rechner verständlichere Form umgewandelt. Solche Programme bekommen Sie, wenn Sie normal mit dem Befehl SAVE"FILENAME" die Programme auf Diskette wegspeichern.

Leider werden diese komprimierten Programme mit dem Befehl MERGE nicht verarbeitet. Um nun doch an einzelne Programmbausteine zu kommen, die sich später miteinander verbinden lassen, müssen alle Programme, die in irgendeiner Form später mit dem Befehl MERGE bearbeitet werden sollen, mit dem Zusatz A für ASCII abgespeichert werden. Dies am Beispiel kurz erläutert:

Sie möchten die Programmteile mit dem Namen "MENUE", "Unterprogramm1" und "Unterprogramm2" einzeln auf Diskette speichern und später beliebig oft in andere Programme einbauen. Es handelt sich also dabei um fertige Programmbausteine, die Sie schon irgendwann erstellt haben. Wenn Sie mit dem Befehl SAVE"MENUE", SAVE"Unterprogramm1" und SAVE"Unterprogramm2" arbeiten, können diese Bausteine später nicht ohne weiteres in andere Programme integriert werden.
Wenn Sie jedoch jedes Programm mit dem Zusatz A abspeichern, dann wird alles, was man so auf dem Bildschirm sieht, in dem gleichen Format auch auf die Diskette gespeichert. Also:

```
SAVE"MENUE",A
SAVE"Unterprogramm1",A
SAVE"Unterprogramm2",A
```

Nun haben Sie Ihre drei Programmbausteine als BASIC-Listing, also im sogenannten ASCII-Format auf der Diskette gespeichert.

Machen Sie mal den Versuch und laden zuerst ein beliebiges Programm von der Diskette. Anschließend holen Sie sich mit `MERGE"MENUE"` den Programmbaustein MENUE in Ihren Arbeitsspeicher. Mit `MERGE"Unterprogramm1"` und `MERGE"Unterprogramm2"` kann man sich die weiteren Bausteine laden. Wichtig ist dabei nur, daß sich die Zeilennummern dabei nicht überschneiden, denn sonst kommt es zu einem heillosen Durcheinander.

Wenn Sie also öfter mit Programmbausteinen arbeiten wollen, dann empfehle ich Ihnen, diese mit dem Zusatz `,A` beim `SAVE`-Befehl abzuspeichern.

Irgendwie muß sich im Betriebssystem der Floppy ein Fehler befinden, weil der Befehl MERGE die normal gespeicherten Programme nicht akzeptiert.

Fehlercode: 25

Bildschirmmeldung:

```
File type error
```

Ursache:

Der Dateityp ist nicht korrekt

Beispiel:

Beim Laden einer Datei müssen Sie wissen, um welchen Dateityp es sich dabei handelt. Ein Dateityp ist z.B. eine ASCII-Datei.

Behebung des Fehlers:

Geben Sie erneut den richtigen Dateityp an und laden erneut.

Fehlercode: 26

Bildschirmmeldung:

```
NEXT missing in Zeilennummer.....
```

Ursache:

Eine Schleife wurde mit FOR zwar geöffnet, doch es fehlt der dazugehörige NEXT-Befehl.

Beispiel:

```
10    DIM WERT(20)
20    FOR I=1 TO 20
30    WERT(I)=I*2
```

Das Programm wird die Fehlermeldung NEXT MISSING IN 20 ausgeben, weil zur Zeile 20 normalerweise noch die Zeile 40 mit dem Befehl NEXT gehört.

Behebung des Fehlers:

Das NEXT als nachträgliche Zeilennummer einfügen.

Fehlercode: 27

Bildschirmmeldung:

```
File already open in Zeilennummer....
```

Ursache:

Eine Datei ist bereits zum Lesen oder zum Schreiben geöffnet worden und reagiert beim zweiten Öffnungsversuch mit dieser Fehlermeldung.

Beispiel:

```
10    DIM D$(100), E$(100)
20    OPENIN"ENGLISCH1"
30    FOR I=1 TO 100
40    INPUT#1, D$(I), E$(I)
50    NEXT I
60    OPENIN"PUNKTESTAND"
70    INPUT#1, PUNKTE
80    CLOSEIN
```

Dieses kleine Programm soll bei einem Vokabeltrainer die Aufgabe übernehmen, 100 Begriffe der Lektion ENGLISCH1 von der Datei zu lesen. Zur Kontrolle des Lernfortschritts soll zusätzlich der letzte Punktestand der Abfragen eingelesen werden.

Da aber eine Datei noch nicht geschlossen wurde (ENGLISCH1), auf der anderen Seite aber die Datei "PUNKTESTAND" eröffnet wird, kommt es zwangsläufig zu dieser Fehlermeldung:

Behebung des Fehlers:

Kontrollieren Sie zuerst einmal, ob irgendwelche Sprungfehler vorliegen, z.B. mit GOTO in einen solche Laderoutine hinein. Ist dies nicht der Fall, muß zuerst die alte Datei mit CLOSEIN oder CLOSEOUT geschlossen werden.

Fehlercode: 28

Bildschirmmeldung:

```
Unknown command
```

Ursache:

Sie haben ein unbekanntes Kommando eingegeben. Das Betriebssystem macht also einen feinen Unterschied zwischen Syntax error und dem Unknown Command.

Meistens handelt es sich dabei um Befehle, die nur im Diskettenbetriebssystem oder bei BASIC-Erweiterungen vorkommen.

Beispiel:

Normalerweise arbeitet der Schneider ja mit dem Datenrecorder als Datenspeicher zusammen. Er ist aber auch in der Lage, spezielle Befehle für die Diskette von anderen zu unterscheiden. Beispielsweise wird auf der Diskette auch das Betriebssystem CP/M vorhanden sein. Jetzt starten wir einmal folgenden Versuch: Schalten Sie das Diskettenlaufwerk einmal ab, falls es an sein sollte und Sie eines haben. Setzen Sie mit <CTRL>, <SHIFT> und <ESC> den Rechner in den Einschaltzustand zurück. Geben Sie nun ein: CPM und drücken die <ENTER>-Taste. Es erscheint ein Syntax error.

Der gleiche Vorgang in leicht veränderter Form. <SHIFT> und <§> drücken und danach CPM eingeben. Nach dem Druck auf die <ENTER>-Taste erscheint nun die Meldung: Unknown command.

Behebung des Fehlers:

Solche Befehle können nur bei entsprechenden Hard- oder Softwareerweiterungen verwendet werden.

Fehlercode: 29

Bildschirmmeldung:

`WEND missing in Zeilennummer.....`

Ursache:

Für die WHILE-Anweisung fehlt die passende WEND-Anweisung.

Beispiel:

Ebenso, wie bei einer FOR...NEXT-Schleife wird die WEND-Anweisung benötigt, um eine solche Schleife zu schließen.

Behebung des Fehlers:

Die WEND-Anweisung als nachträgliche Zeilennummer in ein Programm bringen.

Fehlercode: 30

Bildschirmmeldung:

`Unexpected WEND in Zeilennummer....`

Ursache:

Die Anweisung WEND wurde ohne dazugehörige WHILE-Anweisung gefunden.

Beispiel:

Wie schon oben geschildert, besteht eine Schleife aus einem Befehl zum Öffnen und einem zum Schließen. Wird also eine Schleife geschlossen, muß sie natürlich vorher logischerweise auch geöffnet gewesen sein, sonst erscheint diese Fehlermeldung.

Behebung des Fehlers:

Prüfen, ob ein Sprungbefehl vorliegt. Wenn nicht, die Zeile einfach löschen, in der dieser Fehler auftrat.

18.3 Programmbeispiel: "Deutsche Fehlermeldungen"

Das folgende Programm erzeugt deutsche Fehlermeldungen auf dem Bildschirm, zeigt an, um welchen Fehlercode es sich handelt und in welcher Zeilennummer der Fehler auftrat. Gedacht ist das Programm als Unterprogramm am Ende eines beliebigen anderen Programms. Sinnvollerweise verfahre ich dabei so:

a) Bevor ich mit einem neuen Programm anfange, lade ich mir dieses Programm in den Arbeitsspeicher.

b) Tritt dann beim Programmieren und Ausprobieren ein Fehler auf, kann dieser Fehler sofort analysiert und behoben werden.

Noch eins zur technischen Abwicklung: In dem Programm wird ein Bereich dimensioniert, in dem später die deutschen Fehlermeldungen untergebracht werden. Diese Zeile sollten Sie später irgendwo an den Anfang Ihres Programms setzen, weil diese Zeile nur einmal durchlaufen werden darf. Andernfalls erscheint die Fehlermeldung, daß zum zweiten Mal dimensioniert wird oder zumindest dies versucht wurde.

Das Programm arbeitet mit WINDOW's, wobei sich im Prinzip alles in den unteren Zeilen abspielt. Doch nun zum Programm:

```
59999 END
60000 MODE 2
60010 CLS
60020 INK 0,0
60030 BORDER 0
60040 INK 1,24
60050 PEN 1
60070 WINDOW#1,1,26,20,23
60080 WINDOW#3,28,80,20,23
60081 WINDOW#2,1,80,24,25
60090 PAPER#2,1
60100 CLS#2
60110 PAPER#1,1:PEN#1,0
60120 CLS#1
60130 LOCATE#1,2,2
60140 PRINT#1,"Fehler in Zeile: ";ERL
60150 LOCATE#1,2,3
60160 PRINT#1,"Fehlercode:      ";ERR
60165 fz=ERL
60170 DATA Zu dieser Zeile fehlt noch der Befehl FOR. Bitte kontrollieren.
60180 DATA In der angegebenen Zeilennummer ist ein Syntax-Error enthalten.
60190 DATA Hier steht ein RETURN ohne GOSUB. Vielleicht sind Sie mit GOTO
      falsch eingesprungen? Bitte kontrollieren.
60200 DATA Sie haben in der DATA-Zeile keine Daten mehr zum Lesen. Entweder
      Daten anhängen oder mit RESTORE arbeiten
60210 DATA In dieser Zeile steht ein unerlaubtes Argument. Bitte Zeile listen
      und korrigieren.
60220 DATA ACHTUNG!! Berechnung außerhalb des zulaessigen BASIC-Bereichs.
60230 DATA Arbeitsspeicher voll. Arbeiten Sie am besten mit FRE(1), damit so
      etwas früh genug erkannt wird.
60240 DATA Die Sprunganweisung kann nicht ausgefuehrt werden weil diese
      Zeilennummer nicht existiert.
60250 DATA wahrscheinlich nicht groß genug dimensioniert
60270 DATA Sie haben in dieser Zeile durch 0 geteilt. Achtung!! Folgefehler
      möglich.
60280 DATA Eingabe im Direktmodus nicht moeglich.
60290 DATA Der Variablentyp ist falsch gewaehlt. Bitte aendern.
60300 DATA Achtung! Der String-Bereich ist kompett belegt.
```

```
60310 DATA Zeichenkette leider zu lang!
60320 DATA Die Zeile ist zu kompliziert. Bitte aufteilen
60330 DATA Continue (CONT) leider nicht machbar
60340 DATA Funktion wurde mit DEF FN nicht definiert
60350 DATA RESUME fehlt hier
60360 DATA RESUME liegt nicht innerhalb einer Fehlerroutine
60370 DATA Direktes Kommando gefunden
60380 DATA In dieser Zeile fehlt noch irgendwo eine Angabe zur Berechnung
      oder bei einem Befehl
60390 DATA Diese Zeile ist einfach zu lang
60400 DATA Dateiende überschritte. Bitte EOF als Abfrage beim Laden benutzen
60410 DATA Dateityp nicht in Ordnung
60420 DATA Zum FOR in der angegebenen Zeilennummer fehlt der Befehl NEXT
60430 DATA Achtung! Datei ist bereits eröffnet. Alte Datei erst schließen
60440 DATA Unbekanntes Kommando eingegeben. Haben Sie Ihre Floppy
      angeschaltet?
60450 DATA Zu dieser Zeile fehlt die Anweisung WEND. Bitte eingeben.
60460 DATA WEND-Anweisung ohne WHILE gefunden. Bitte kontrollieren.
60500 DIM fehlermeldung$(30)
60510 FOR i=1 TO 30
60520 READ fehler$
60530 fehlermeldung$(i)=fehler$
60540 NEXT i
60550 RESTORE 60170
60590 PEN#2,0
60600 LOCATE#2,1,1:PRINT#2,"Meldung: ";fehlermeldung$(ERR);
60610 LOCATE#3,2,1:PRINT#3,"Jetzt können Sie folgendes tun:"
60620 LOCATE#3,2,2:PRINT#3,"1 = weitermachen   2 = Programm beenden"
60640 LOCATE#3,2,4:PRINT#3,"Was wollen Sie:   (1 oder 2)? "
60650 a$=INKEY$:IF a$="" THEN 60650
60660 IF a$="1" THEN ERASE fehlermeldung$:RESUME
60670 IF a$="2" THEN GOTO 60900
60680 SOUND 1,100,200,7
60690 SOUND 2,101,200,7
60700 GOTO 60650
60900 MODE 2:WINDOW#1,1,80,24,25
60910 CLS#1
60920 PRINT#1,"Sie können jetzt die fehlerhafte Programmzeile ";ERL;"
      verbessern
60930 PRINT#1,"und mit EDIT";ERL;" oder LIST";ERL;" bearbeiten."
60935 LOCATE 1,1
60940 END
```

Zum Programm läßt sich noch sagen, daß wegen der Länge der Fehlermeldungen nicht immer grammatisch richtig im dafür vorgesehenen Fenster getrennt wird. Dafür haben Sie aber eine recht übersichtliche Angabe über Fehlerart und Zeilennummer, wo der Fehler auftrat.

Betrachten Sie sich einmal die Zeilennummer 60170. Hier steht FZ=ERL, also eine Variablenzuweisung. Dabei sollte die Fehlerzeilennummer in einer Variablen gespeichert werden. Damit hatte ich später vor, diese Zeilennummer mit LIST FZ oder EDIT FZ zu bearbeiten.

Leider wird dies nicht vom System unterstützt, wie bei manchen anderen Computern.

Sie können das Programm "Deutsche Fehlermeldungen" natürlich auch voll in Ihre eigenen Programme integrieren, wenn Sie immer im MODE 2 arbeiten und unten am Bildschirm die nötigen Zeilen frei lassen, die dieses Unterprogramm benötigt. Dabei können dann auch die MODE-Anweisungen und die Befehle CLS, BORDER usw. wegfallen. Falls die Fehlermeldung am unteren Bildschirmrand immer noch nicht zum gewünschten Erfolg führt, sollten Sie sich den Text über den Fehlercode auf den letzten Seiten noch einmal durchlesen.

19 Fertig programmierte Programmbausteine

Bei der Erstellung von neuen Programmen gibt es immer wieder Programmteile, die sich ständig wiederholen. Meistens kommt noch dazu, daß solche Programmteile innerhalb eines Programms auch noch mehrfach in Erscheinung treten. Beispielsweise könnten dies Abfragen oder Prüfungen, Warteschleifen, Menüs, Fehlermeldungen, Melodien oder andere Dinge sein, die man eigentlich immer wieder gebrauchen kann.

Bedenken Sie, daß Sie das Rad ja nicht jedesmal neu erfinden müssen!

Natürlich kostet die Programmierung auch Zeit und zwar mehr, als man es sich am Anfang vorstellt. Wenn Sie sich immer wieder mit Routineaufgaben aufhalten müssen, die Sie ja schon einmal irgendwann programmiert haben, kommen Sie vorläufig jedenfalls nicht ans Ziel.

Um sich aber speziell auf ein bestimmtes Problem einzustellen, sollten die Routinetätigkeiten aber so weit wie möglich eingeschränkt werden.

Eine besonders praktische Lösung wäre es doch, wenn man auf bereits erstellte Programmbausteine zurückgreifen könnte und dann nur noch die gewünschten Bausteine zusammensetzen brauchte.

Diese Möglichkeit besteht auch beim Schneider CPC 464.

Wenn man einzelne Programmbausteine auf Kassette oder Diskette speichert, können diese Bausteine später mit dem BASIC-Befehl MERGE einfach miteinander verbunden werden. Vor Beginn eines neuen Programms laden Sie als erstes alle nötigen Routineabfragen und alle sonstigen Bausteine, die Sie brauchen in den Rechner. Dann fangen Sie an mit der Programmierung des eigentlichen Problems und haben so schon eine ganze Menge Arbeit und Zeit gespart.

Wichtig bei der ganzen Sache ist nur, daß jeder Baustein einen anderen Zeilennumernbereich hat, sonst kommt es später zu Schwierigkeiten beim Nachladen anderer Programmteile. Haben beispielsweise zwei Unterprogramme denselben Bereich der Zeilennummern, wird beim Nachladen des nächsten Moduls der alte Bereich im Arbeitsspeicher des Computers mit

den Zeilennummern des neuen Bausteins überschrieben. Solche Fehler sind dann später auf den ersten Blick nicht so leicht zu erkennen.

Am besten ist es, wenn Sie sich eine Liste anlegen, in der alle Bausteine mit Zeilennummernbereichen aufgeführt sind. Sie haben so auf den ersten Blick eine Übersicht, welche Zeilenbereiche schon belegt sind und welche noch nicht.

Bei Bedarf sollten Sie auch einen Baustein mit dem Befehl **RENUM** umnumerieren, damit er in Ihr Schema paßt.

Auf den nächsten Seiten finden einige Programmbausteine, die aus meiner Programmbibliothek stammen. Alle Bausteine sind funktionstüchtig und werden einfach mit dem Befehl **GOSUB** `Zeilennummer` aufgerufen. Damit sind schon einige Routinearbeiten vom Tisch und außerdem können Sie sich bei der Fehlersuche in neu entwickelten Programmen auf den wesentlichen Teil konzentrieren, da alle Bausteine von mir fehlerfrei arbeiten.

Leider sind Tippfehler bei der Eingabe nie ganz auszuschließen und es mal vorkommen kann, daß sich trotzdem bei Ihnen ein Fehler einschleicht.

Falls Sie also einmal ein neues Programm anfangen, sehen Sie sich am besten einmal auf den nächsten Seiten um, ob vielleicht schon etwas passendes dabei ist. Dazu sind diese Bausteine ja schließlich da.

20 Vom Problem zur Lösung - Software-Entwicklung am praktischen Beispiel

Als Anfänger wird man sich (jetzt mal ein wenig übertrieben) an den Computer setzen und einfach munter lostippen, frei nach dem Motto, es wird wohl schon etwas dabei herauskommen. Und wirklich, es kommt auch etwas dabei heraus. Nur ist dies bestimmt nichts gutes.

Wenn Sie schon einmal ein paar BASIC-Programme geschrieben haben, die länger als vielleicht 2 oder 3 Bildschirmseiten sind, holen Sie sich diese nochmal zum Vergleich heran. Mit Sicherheit wimmelt es in Ihren Programmen von **GOTO**-Befehlen und mit den REM-Anweisungen haben Sie bestimmt auch recht gut gespart.

Solche Dinge sind bei fast allen Anfängern immer gleich. Im Prinzip ist das nichts schlimmes, wenn man sich diesen "wilden" Programmierstil im Lauf der Zeit wieder abgewöhnt.

Bei kleinen Programmen macht sich der schlechte Programmstil nicht so gravierend bemerkbar. Aber spätestens, wenn es um größere Programme, wie z.B. eine Textverarbeitung oder Dateiverwaltung geht, kommt unweigerlich die große Pleite.

Diese Pleite werden auch Sie mit Sicherheit irgendwann erleben. Und das ist auch ganz gut so, denn dann gewöhnen Sie sich schon von selbst eine andere Arbeitsweise beim Programmieren an.

In diesem Kapitel sollen Sie also lernen, wie man größere Softwareprojekte plant und in die Tat umsetzt.

Dazu habe ich mir eine Methode zur Softwareentwicklung erarbeitet, nach der ich alle Projekte plane und durchführe. Diese Methode hat sich bei mir seit mehreren Jahren bestens bewährt. Beispielsweise erstelle ich nach dieser Methode spezielle EDV-Lösungen für die Industrie.

Ausgangssituation

Problem

Lösung

Methode zur Softwareentwicklung

Start

Planungsphase

Problem definieren — IST-Zustand aufnehmen

Ziele setzen — SOLL-Zustand beschreiben

Aufgabe abgrenzen — Module bilden

Lösungen suchen — nach allen denkbaren Lösungsansätzen suchen

Durchführungsphase

Beste Lösung weiterentwickeln

EDV-Lösung einführen — Zielerfüllung kontrollieren

Ziel

Bild 20.1:

Kurz zur Beschreibung:

Sie können diese Methode grob gesehen in zwei Phasen teilen. Die erste Phase ist die sogenannte Planungsphase und für mich persönlich immer wieder die interessanteste, weil es oft verschiedene Lösungswege gibt, die zum Ziel führen. Sie können also in dieser Phase Ihre Kreativität unter Beweis stellen.

Die zweite Phase bezeichne ich als Durchführphase. Hier wird aus der Planung heraus eine konkrete Aufgabe gestellt, die nun ausgeführt werden muß. Zu dieser Phase gehört z.B. die Fehlersuche, Freigabe und Praxis-Programmierung, Einführung usw.

Als Problemstellung habe ich hier die Rechnungsschreibung gewählt, weil dieses Beispiel für jeden leicht verständlich ist und auch einen entsprechenden Umfang aufweist.

Die einzelnen Stufen im Detail:

Stufe 1: Problem definieren, IST-Zustand erfassen

Das Problem: Ein Geschäft für elektronische Bauteile verschickt laufend Rechnungen mit diversen Rechnungspositionen. Bisher schrieb der Inhaber am Wochenende seine Rechnungen, wobei bei einem Rechenfehler alle anderen Positionen korrigiert werden mußten.

Folgende Daten gab er für die Softwarelösung an:

Anzahl Rechnungen pro Woche: 25
Anzahl Rechnungspositionen: ca. 10

Da neben den elektronischen Bauteilen auch Fachzeitschriften und Bücher verschickt werden, muß er mit zwei Mehrwertsteuersätzen arbeiten. Zur Zeit handelt es sich dabei um 7 und 14% Mehrwertsteuer.

Stufe 2: Zielsetzung

Bei der Zielsetzung spielen Hard- und Software eine gleichbedeutende Rolle. Da Sie ja sicherlich den Schneider besitzen, wollen wir den Punkt Hardware einmal ausschließen. Bleibt also die Software übrig.

Dafür gilt es also jetzt, Ziele festzulegen. Da in dem Geschäft für elektronische Bauteile auch andere Mitarbeiter einmal die Rechnungen schreiben sollen, war das erste konkrete Ziel die leichte Bedienbarkeit der Softwarelösung.

Das zweite Ziel war die Datenspeicherung nach Erstellung der Rechnung auf einem Externspeicher.

Das dritte Ziel ist ein einheitlicher Aufbau von Datensätzen, um spätere Auswertungen zu unterstützen.

Als viertes Ziel wurde vereinbart, daß eine Datenschnittstelle zur Adressverwaltung vorhanden sein soll, um späteren Entwicklungen Sorge zu tragen.

Zusätzlich wurde als fünftes Ziel vereinbart, daß unter der Rechnung ein beliebiger Text erscheinen sollte (z.B. Zahlungsbedingungen, Werbetexte).

Stufe 3: Aufgaben abgrenzen, Teilaufgaben bilden

Teilaufgabe 1	Benutzerführung
Teilaufgabe 2	Dateneingabe
Teilaufgabe 3	Rechnung laden
Teilaufgabe 4	Rechnung speichern
Teilaufgabe 5	Bildschirmausgabe
Teilaufgabe 6	Kundenanschrift laden
Teilaufgabe 7	Druckerausgabe
Teilaufgabe 8	Programm beenden

Dieses ist mit der wichtigste Schritt der gesamten Methode. Es ist unmöglich, daß sich ein einziger Programmierer gleichzeitig auf jede Funktion stürzen kann. Hier muß also konsequent eine Aufgabe durchgeführt werden, bevor man die nächste Aufgabe angeht. Teilaufgaben oder Module zu bilden, ist in vielen Fällen nicht gerade einfach. Man bekommt aber im Laufe der Zeit einen Blick dafür, wo sich ein Modul bilden läßt und wo nicht.

Als Grundsatz kann man davon ausgehen: Je kleiner die Teilaufgabe, desto leichter läßt sich später das Gesamtprojekt bearbeiten.

Stufe 4: Alle nur denkbaren Lösungsansätze prüfen.

Hier in Stufe 4 werden also verschiedene Ansätze geprüft, ob sie sich für dieses spezielle Projekt eignen oder nicht.

Folgende Lösungsansätze sind denkbar:

a) Programmerstellung durch Ähnlichkeitsvergleich (ist schon ein ähnliches Programm erstellt worden)
b) Programmerstellung durch externe Mitarbeiter

c) Programmerstellung durch Einsatz von Programm-Bausteinen
d) Programmerstellung durch Programmgenerator
e) Programmerstellung durch völlige Neukonzeption
f) Suche nach Standardpaketen für dieses Problem

Stufe 5: Besten Lösungsweg weiterentwickeln

Zur Vorgehensweise in diesem Schritt: Für jede Teilaufgabe kann im Prinzip eine der aufgeführten Lösungsansätze in Frage kommen. Für die optimalste Lösung halte ich den Einsatz von Programmbausteinen, weil diese schon oft vorhanden sind und sich daher viel Zeit beim Programmieren sparen läßt. Da aber nicht für alle Teilaufgaben fertige Programmbausteine auf Lager sind, werden wir unser Softwareprojekt kombinieren aus Neuentwicklungen und Programmbausteinen. Zu Stufe 5 gehört auch das Austesten des Programms mit Hilfe von Testdaten.

Stufe 6: Lösung einführen und Zielerfüllung kontrollieren

Neben der Planung des Projekts ist das eigentlich einer der wichtigsten Schritte, denn hier wird sich zeigen, wie gut oder wie schlecht dieses Projekt geplant wurde. Zur Einführung gehört der Aufbau des kompletten Computersystems incl. der nötigen Software.

Ein Punkt, der oft bei diesen Dingen vernachlässigt wird, ist die Einarbeitung der Personen, die später am Computer arbeiten müssen. Es reicht also nicht aus, einen Computer mit Software zu kaufen und anschließend nur noch den Stecker in die Steckdose zu stecken. Gerade im Bereich der Mitarbeiterschulungen wird sich aber in den nächsten Jahren einiges tun müssen, der Trend zu mehr EDV steigt seit Jahren unaufhaltsam.

Ebenfalls zur Einführung gehört der Aufbau eines kompletten Datenbestands, mit dem später gearbeitet werden soll. Auch hier sollte man genaustens kontrollieren, damit man vor unliebsamen Überraschungen sicher ist.

Für uns ist speziell die Stufe 5 in Verbindung mit Stufe 3 interessant. Dabei geht es um die Programmierung der Teilaufgaben.

Fangen wir einmal mit der einfachsten Teilaufgabe an, der Benutzerführung. Als Ziel wurde ja festgelegt, daß das Programm benutzerfreundlich sein sollte. Dazu können wir uns der schon bekannten Menü-Technik bedienen. Weil uns ein solches Programm schon als Baustein vorliegt, brauchen wir das Menü nur noch mit den entsprechenden Texten zu versehen:

```
1 = Dateneingabe
2 = Rechnung laden
3 = Rechnung speichern
4 = Bildschirmausgabe
5 = Kundenanschrift laden
6 = Druckerausgabe
7 = Programm beenden
```

Tauschen Sie also die Bezeichnungen in DATA-Zeilen gegen die neuen Bezeichnungen aus.

Programmlisting Baustein 1

```
100    DATA 1,Rechnung erstellen
110    DATA 2,Rechnung laden
120    DATA 3,Rechnung speichern
130    DATA 4,Ausgabe auf Bildschirm
140    DATA 5,Kundenanschrift laden
150    DATA 6,Ausgabe auf Drucker
160    DATA 7,Programm beenden
170    REM
180    REM ****** Datenfelder dimensionieren******
190    REM
200    z%=20:DIM mwst$(z%),mwst(z%),ep(z%),menge(z%),summe(z%)
210    DIM artnr$(z%),artbez$(z%),ep$(z%),menge$(z%)
220    REM
230    REM ******** Datenfelder dimensioniert******
240    REM
250    MODE 1:CLS:INK 0,0:BORDER 0:i$=CHR$(24)
260    GOSUB 380:'*** MENÜ AUFBAUEN*****
270    a$=INKEY$:IF a$="" THEN 270
280    z=VAL (a$)
290    IF z<1 OR z>7 THEN SOUND 1,100,20,7:GOTO 270
300    IF a$="1" THEN CLEAR:GOSUB 1000
310    IF a$="2" THEN CLEAR:GOSUB 2000
320    IF a$="3" THEN GOSUB 3000
330    IF a$="4" THEN GOSUB 4000
340    IF a$="5" THEN GOSUB 5000
350    IF a$="6" THEN GOSUB 6000
360    IF a$="7" THEN GOSUB 7000
370    GOTO 260
380    '******* Menue aufbauen *********
390    BORDER 0:INK 0,0:INK 1,24:MODE 1.CLS:MOVE 0,0
400    MODE 1:CLS:MOVE 0,0:DRAW 0,399:DRAW 639,399
410    DRAW 0,399:DRAW 639,399:DRAW 639,0:MOVE 0,0
420    DRAW 639,0:i=350:MOVE 0,i:DRAW 639,i
430    FOR i=307 TO 0 STEP -49:MOVE 0,i:DRAW 639,i: NEXT
440    FOR i=15 TO 0 STEP -1:MOVE 0,i:DRAW 639,i:NEXT
450    MOVE 100,0:DRAW 100,399-50:LOCATE 2,2:PEN 2
460    PRINT" H A U P T - M E N U E":PEN 1
470    '****** Menue bezeichnen ******
480    RESTORE
490    FOR i=5 TO 23 STEP 3
500    READ a$,B$
510    LOCATE 3,i:PRINT a$
```

```
520   LOCATE 10,i:PRINT b$
530   NEXT i
540   REM ****** Menue fertig ******
550   RETURN
```

Programmbeschreibung für Baustein 1 - Menue und Dimensionierung

Zeilen 100-160: DATA-Angaben für die einzelnen Bezeichnungen im Menü

Zeilen 170-190: Kommentárzeilen

Zeilen: 200-210: Dimensionierung der einzelnen Variablen mit 20 Feldern. Folgende Variablen wurden dabei verwendet:
MWST$ und MWST = Mehrwertsteuer in DM. Im ersten Fall liegt die Mehrwertsteuer als Text vor, im zweiten Fall als Zahl.
EP$ und EP = Einzelpreis in DM,
MENGE$ und MENGE = Anzahl gleicher Artikel.
ARTNR$ = Artikelnummer
ARBEZ$ = Artikelbezeichnung
SUMME = Einzelpreis * Menge

Zeile 220-240: Kommentarzeilen

Zeile 250: Bildschirmmodus wählen, Farbe setzen. Der Variablen I$ wird der ASCII-Code für die REVERS-Funktion zugewiesen

Zeile 260: Unterprogramm "Menue" aufrufen

Zeile 270: Abfrage Tastatur

Zeile 280: Die gedrückte Taste wird als Zeichen umgewandelt in einen Text

Zeile 290: Liegt der umgewandelte Wert unter 1 oder über 7, gibt das Programm eine akustische Meldung aus und verzweigt wieder zurück zur Tastaturabfrage

Zeile 300 bis 360: Die entsprechenden Unterprogramme werden mit GOSUB aufgerufen

Zeile 370: Rücksprung aus dem Unterprogramm. Menue wird erneut aufgebaut

Zeile 380-460: Komplettes Menue wird gezeichnet

Zeile 470: Kommentarzeile

Zeile 480: DATA-Zeiger wieder an den Anfang der ersten DATA-Position setzen.

Zeile 490-540: Bezeichnungen für das Menue aus den ersten DATA-Zeilen lesen und im Menue positioniert ausdrucken

Zeile 550: Rücksprung aus dem Unterprogramm mit RETURN

Programmlisting Baustein 2

```
1000  REM ******Dateneingabe ******
1005  GOSUB 1800:REM  neu dimensionieren
1010  MODE 2:CLS:WONDOW#1,1,80,1,2:PAPER#1,1
1020  cls#1:pen#1,0
1030  modus$="Modus:  Dateneingabe"
1040  LOCATE#1,2,2:PRINT#1,modus$
1050  WINDOW#2,1,80,5,22:CLS#2:PEN#2,1
1060  maximal=0
1070  FOR i=1 TO 100
1080  LOCATE#2,1,2
1090  PRINT#2,"Dateneingabe beenden,";
1100  PRINT#2,"dann bei Art.-Nr. ende eingeben...."
1110  LOCATE#2,1,4:PRINT#2,"Lfd.-Nr.";i
1120  LOCATE#2,1,5:PRINT#2,"Artikel-Nr.:  "
1130  LINE INPUT#2,"",artnr$(i)
1140  zeiger=0
1150  IF artnr$(i)="ende" OR artnr$(i)="Ende" THEN zeiger=1
1160  If zeiger=1 THEN maximal=i-1:GOTO 1490
1170  LOCATE#2,1,6:PRINT#2,"Artikel-Bez.:  ";
1180  LINE INPUT#2,"",artbez$(i)
1190  LOCATE#2,1,7:PRINT#2,"Einzel-Preis: ";
1200  LINE INPUT#2,"",ep$(i)
1210  LOCATE#2,1,8:PRINT#2,"Bestell-Menge: ";
1220  LINE INPUT#2,"",menge$(i)
1230  LOCATE#2,1,9
1240  PRINTER#2,"MWST ? 1 = 7 %  2= 14 % ";
1250  LINE INPUT#2,"",mwst$(i)
1260  p=0
1270  IF mwst$(i)="1" THEN p=7:GOTO 1300
1280  IF mwst$(i)="2" THEN p=14:GOTO 1300
1290  SOUND 1,200,100,7:GOTO 1230
1300  ep(i)=VAL(ep$(i))
1310  menge(i)=VAL(menge$(i))
1320  summe(i)=ep(i)*menge(i)
1330  brutto=0
1340  brutto=summe(i)*(100+p)/100
1350  mwst(i)=brutto-summe(i)
1360  mwst(i)=INT(mwst(i)*100+0.5)/100
1370  IF p=7 THEN mwst1=mwst1+mwst(i)
1380  IF p=14 THEN mwst2=mwst2+mwst(i)
1390  mwges=mwst1+mwst2
1400  gesamtsumme=gesamtsume+summe(i)
1410  LOCATE#2,1,12
1420  PRINT#2,"Daten so richtig eingegeben ? ";
1430  PRINT#2," j=ja n=nein "
1440  a$=INKEY$:IF a$"" THEN 1440
1450  IF a$="j" THEN CLS#2:GOTO 1480
1460  IF a$="n" THEN CLS#2:GOTO 1080
1470  SOUND 1,200,50,7:SOUND 2,201,50,7:GOTO 1440
1480  NEXT
1490  endbetrag=mwges+gesamtsumme
1500  REM **** zurück ins Menü ******
1510  RETURN
1800  z%=20:DIM mwst$(z%),mwst(z%),ep(z%),menge(z%),summe(z%)
1810  DIM artnr$(z%),artbez$(z%),ep$(z%),menge$(z%)
```

```
1820  REM bereit für Neueingabe
1830  RETURN
```

Programmbeschreibung für Baustein 2 - Dateneingabe

In diesem Baustein wird die Dateneingabe behandelt und bearbeitet. Es handelt sich im einzelnen um folgende Angaben:

a. Artikel-Nummer
b. Artikel-Bezeichnung
c. Einzelpreis
d. Bestellmenge
e. Mehrwertsteuer-Satz

Sollten Ihnen bei der Eingabe der Daten ein Fehler unterlaufen, können Sie Ihre Eingaben selbstverständlich korrigieren. Das Programm gibt nach erfolgter Eingabe die Nachricht auf den Bildschirm "Daten so richtig ?". Wenn Sie dann die Taste J betätigen, wird die Eingabe als korrekt angesehen und es wird nach dem nächsten Artikel gefragt.

Beenden können Sie diesen Modus, wenn Sie bei der Artikel-Nummer einfach das Wort *ende* eingeben, wobei Sie den Begriff klein schreiben sollten.

Festgelegt wurde schon im Baustein, daß maximal 20 Artikel bearbeitet werden können, weil ja auch sonst kein Platz mehr auf der Rechnung ist.

Zeile 1000: Kommentarzeile
Zeile 1010: Modus 2 wählen, Bildschirm löschen, WINDOW in den ersten beiden Zeilen erzeugen, WINDOW-Farbe setzen mit dem Befehl PAPER
Zeile 1020: WINDOW löschen und Schriftfarbe schwarz für dieses Fenster wählen
Zeile 1030: Modus bestimmen
Zeile 1040: Textcursor in WINDOW#1 positionieren und den Modus ausgeben
Zeile 1050: WINDOW#2 definieren, dieses Fenster dient später zur Dateneingabe
Zeile 1060: Die Variable MAXIMAL auf Null setzen
Zeile 1070: Schleife öffnen für die Eingabe Artikel
Zeile 1080: Textcursor im WINDOW#2 positionieren
Zeile 1090: Textausgabe
Zeile 1100: Textausgabe
Zeile 1110: Laufende Nummer ausgeben, geschieht automatisch
Zeile 1120: Textausgabe

Zeile 1130: Eingabe der Artikel-Nummer, wobei alphanumerische Zeichen möglich sind
Zeile 1140: Wenn bei der Artikel-Nummer das Wort "ende" oder "Ende" eingegeben wurde, dann den Endbetrag in Zeile 1480 berechnen.
Zeile 1150: Textausgabe
Zeile 1160: Eingabe der Artikel-Bezeichnung
Zeile 1170: Textausgabe
Zeile 1180: Eingabe des Einzelpreises in DM/Netto
Zeile 1190: Textausgabe
Zeile 1200: Eingabe der bestellten Menge in Stück
Zeile 1210: Textcursor neu positionieren
Zeile 1220: Textausgabe, welcher Steuersatz möglich ist
Zeile 1230: Eingabe des Steuersatz
Zeile 1240: Variable P auf Null setzen. P dient zur Speicherung des Steuerkennzeichens.
Steuerkennzeichen 1 bedeutet 7% Mehrwertsteuer
Steuerkennzeichen 2 bedeutet 14% Mehrwertsteuer
Zeile 1250: Ist das Kennzeichen 1 eingegeben worden, dann P auf 7% setzen und zur Zeile 1280 gehen
Zeile 1260: Ist das Kennzeichen 2 eingegeben worden, dann P auf 14% Mehrwertsteuer setzen und zur Zeile 1280 gehen.
Zeile 1270: Akustisches Signal für Fehleingabe ausgeben und den Steuersatz erneut abfragen
Zeile 1280: Den Einzelpreis als Text in eine reine numerische Eingabe umwandeln, mit der man auch rechnen kann
Zeile 1290: Menge wie den Einzelpreis behandeln
Zeile 1300: Nettopreis pro Artikel berechnen
Zeile 1310: Die Variable BRUTTO bei jedem Schleifendurchlauf auf Null setzen
Zeile 1320: Den Bruttobetrag für diese Menge von Artikeln berechnen
Zeile 1330: Die dazu passende Mehrwertsteuer berechnen
Zeile 1340: Die Steuer in DM auf zwei Nachkommastellen berechnen
Zeile 1350: Wenn P = 7 ist, dann den eben berechneten Betrag zur Summe mit 14 % addieren
Zeile 1360: Wenn P = 14 ist, dann wie bei 7 % verfahren, nur jetzt zur Summe mit 14 % addieren
Zeile 1370: Mehrwertsteuergesamtsatz setzt sich aus den beiden Summen MWST1 und MWST2 zusammen.
Bei jedem Durchlauf der Schleife wird dieser Gesamtsteuerbetrag aufaddiert

Zeile 1380: Nettosumme laufend aufaddieren. Diese Nettosumme wird hier als GESAMTSUMME bezeichnet
Zeile 1390: Textcursor positionieren
Zeile 1400: Textausgabe "Daten so richtig eingegeben ?"
Zeile 1410: Textausgabe "j = ja   n = nein
Zeile 1420: Tastaturabfrage
Zeile 1430: Wenn der Buchstabe j gedrückt wird, dann den Bildschirm (WINDOW#2) löschen und zur Zeile 1460 gehen
Zeile 1440: Wird n eingegeben, dann wird die gesamte Eingabe wiederholt
Zeile 1450: Akustisches Signal bei Falscheingabe ausgeben und in Zeile 1420 erneut die Tastatur abfragen
Zeile 1460: Anzahl der Eingaben um eins erhöhen
Zeile 1470: Schleife zur Dateneingabe wieder abschließen
Zeile 1480: Bruttobetrag errechnen
Zeile 1490: Kommentarzeile
Zeile 1500: Rücksprung aus dem Unterprogramm mit RETURN

Programmlisting Baustein 3

```
2000  modus$="Modus: Rechnung laden "
2005  GOSUB 2800:REM  neu dimensionieren
2010  MODE 2:CLS:i$=CHR$(24)
2020  WINDOW#1,1,80,1,2:PAPER#1,1:CLS#1:PEN#1,0
2030  WINDOW SWAP 0,2
2040  WINDOW#0,1,80,4,23:PAPER#0,0:CLS#0:PEN#0,1
2050  WINDOW#2,1,80,4,23:PAPER#2,0:CLS#2:PEN#2,1
2060  WINDOW#3,1,80,25,25:PAPER#3,1:CLS#3:PEN#3,0
2070  PRINT#3,"Meldungen:";
2080  PEN#1,0:LOCATE#1,2,1:PRINT#1,modus$
2090  '******* Rechnung laden *******
2100  LOCATE#2,1,4:PRINT#2,"Bitte eine Diskette einlegen ";
2110  PRINT#2,"und anschliessend eine Taste drücken...."
2120  a$=INKEY$:IF a$="" THEN 2120
2130  LOCATE#2,1,4:PRINT#2,STRING$(78,32)
2140  LOCATE#2,1,4:PRINT#2,"Auf Ihrer Diskette befinden ";
2150  PRINT#2,"sich folgende  Einträge:"
2160  LOCATE#0,1,5
2170  CAT
2180  PRINT#3," Directory der Diskette geladen....."
2190  PRINT#0
2200  LINE INPUT#0,"Welche Rechnung möchten Sie laden: ",la$
2210  IF LEN(la$)>8 THEN GOSUB 2560:GOTO 2190
2220  OPENIN la$
2230  LINE INPUT#9,date$
2240  LINE INPUT#9,rnr$
2250  LINE INPUT#9,anrede$
2260  LINE INPUT#9,name1$
2270  LINE INPUT#9,name2$
2280  LINE INPUT#9,strasse$
2290  LINE INPUT#9,ort$
```

```
2300  LINE INPUT#9,text1$
2310  LINE INPUT#9,text2$
2320  INPUT#9,gesamtsumme
2330  INPUT#9,mwst1
2340  INPUT#9,mwst2
2350  INPUT#9,mwges
2360  INPUT#9,endbetrag
2370  FOR i=1 TO 100
2380  zeiger=0
2390  IF EOF THEN SOUND 1,100,100,7:zeiger=1
2400  IF zeiger=1 THEN LOCATE#3,13,1
2410  IF zeiger=1 THEN PRINT#3,"Rechnung ist geladen......"
2420  IF zeiger=1 THEN GOTO 2530
2430  zeiger=0
2440  INPUT#9,artnr$(i)
2450  INPUT#9,artbez$(i)
2460  INPUT#9,ep(i)
2470  INPUT#9,menge(i)
2480  INPUT#9,mwst(i)
2490  INPUT#9,summe(i)
1200  maximal=maximal+1
2510  NEXT i
2520  '**** Datei wieder schließen ******
2530  CLOSEIN
2540  '******* Rechnung speichern *****
2550  RETURN
2560  PRINT#2,"Name zu lang. bitte neu eingeben."
2570  RETURN
2800  z%=20:DIM mwst$(z%),mwst(z%),ep(z%),menge(z%),summe(z%)
2810  DIM artnr$(z%),artbez$(z%),ep$(z%),menge$(z%)
2820  REM  bereit für Neueingabe
2830  RETURN
```

Programmbeschreibung für Baustein 3 - Rechnung laden

Mit diesem Baustein ist es möglich, erstellte und abgespeicherte Rechnungen jederzeit wieder in den Arbeitsspeicher des Computers zu laden.

Das Programm lädt dabei folgende Daten:

a. das Rechnungsdatum
b. die Rechnungsnummer
c. die Anrede des Kunden
d. den ersten Kundennamen
e. den zweiten Kundennamen
f. die Straße und Hausnummer oder das Postfach
g. den Ort mit Postleitzahl
h. zwei Zeilen mit Werbetexten
i. den Gesamtnettobetrag
j. den Mehrwertsteuersatz zu 7 %
k. den Mehrwertsteuersatz zu 14 %
l. den Gesamtbetrag der Mehrwertsteuer

m. den Endbetrag (Brutto)

Anschließend werden mit Hilfe einer Schleife folgende Daten eingelesen:

n. die Artikel-Nummer
o. die Artikel-Bezeichnung
p. der Einzelpreis
q. die bestellte Menge
r. die Mehrwertsteuer
s. die Nettosumme

Zeile 2000: Modus festlegen
Zeile 2010: Bildschirm auf MODE 2 umschalten, löschen und die Reversdarstellung der Variablen I$ zuweisen.
Zeile 2020-2060: WINDOW 0 bis 3 definieren
Zeile 2070: Text auf dem 3. Fenster ausgeben
Zeile 2080: Schriftfarbe wählen (schwarz) und Textcursor neu positionieren
Zeile 2090: Kommentarzeile
Zeile 2100-2110: Textausgabe positioniert auf WINDOW 2
Zeile 2120: Abfrage der Tastatur
Zeile 2130: Den Text in WINDOW 2 mit 78 Leerzeichen überschreiben
Zeile 2140-2150: Textcursor neu positionieren und Text ausgeben
Zeile 2160: Textcursor in WINDOW 3 ausgeben
Zeile 2170: Inhaltsverzeichnis der Diskette ansehen
Zeile 2180: Nachricht in WINDOW 3 ausgeben
Zeile 2190: Leerzeile in WINDOW 0 ausgeben
Zeile 2200: Eingabe, welche Rechnung geladen werden soll
Zeile 2210: Wenn die Länge der Zeichen mehr als acht beträgt, dann Unterprogramm aufrufen und Eingabe wiederholen
Zeile 2220: Datei zum Einlesen der Daten öffnen
Zeile 2230-2520: Datei komplett einlesen
Zeile 2530: Datei wieder schließen
Zeile 2540: Kommentarzeile
Zeile 2550: Rücksprung ins Hauptprogramm mit RETURN
Anmerkung: Der Zähler MAXIMAL zählt die Anzahl der gespeicherten Artikel. Siehe Zeile 2500
Zeile 2560-2570: Text ausgeben, daß der Name zu lang ist

Programmlisting Baustein 4

```
3000 modus$="Modus: Rechnung speichern"
3010 MODE 2:CLS:i$=CHR$(24)
3020 WINDOW#1,1,80,1,2:PAPER#1,1:CLS#1:PEN#1,0
```

```
3030  WINDOW SWAP 0,2
3040  WINDOW#0,1,80,4,23:PAPER#0,0:CLS#0:PEN#0,1
3050  WINDOW#2,1,80,4,23:PAPER#2,0:CLS#2:PEN#2,1
3060  WINDOW#3,1,80,25,25:PAPER#3,1:CLS#3:PEN#3,1
3070  PRINT#3," Meldungen:";
3080  PEN1,0:LOCATE#1,2,1:PRINT#1,modus$
3090  '********* Rechnung speichern *****
3100  IF macimal=0 THEN GOSUB 3510:GOTO 3490
3110  LOCATE#2,1,4:PRINT#2,"Bitte eine Diskette einlegen ";
3120  PRINT#2,"und anschließend eine Taste druecken...."
3130  a$=INKEY$:IF a$="" THEN 3130
3140  LOCATE#2,1,4:PRINT#2,STRING$(78,32)
3150  LOCATE#2,1,4:PRINT#2,"Auf Ihrer Diskette";
3160  PRINT#2,"befinden sich folgende Einträge:"
3170  LOCATE#0,1,5
3180  CAT
3190  PRINT#3," Directory der Diskette geladen....."
3200  PRINT#0:PRINT#0,"Mit welchem Namen wollen ";
3210  LINE INPUT#0,"Sie die Rechnung speichern: ",la$
3220  IF LEN(la$)>8 THEN GOSUB 3540:GOTO 3200
3230  OPENOUT la$
3240  PRINT#9,date$
3250  PRINT#9,rnr$
3260  PRINT#9,anrede$
3270  PRINT#9,name1$
3280  PRINT#9,name2$
3290  PRINT#9,strasse$
3300  PRINT#9,ort$
3310  PRINT#9,text1$
3320  PRINT#9,text2$
3330  PRINT#9,gesamtsumme
3340  PRINT#9,mwst1
5350  PRINT#9,mwst2
3360  PRINT#9,mwges
3370  PRINT#9,endbetrag
3380  FOR i=1 TO maximal
3390  PRINT#9,artnr$(i)
3400  PRINT#9,artbez$(i)
3410  PRINT#9,ep(i)
3420  PRINT#9,menge(i)
3430  PRINT#9,mwst(i)
3440  PRINT#9,summe(i)
3450  NEXT i
3460  CLOSEOUT
3470  LOCATE#3,13,1:PRINT#3,"Die Rechnung wurde ";
3480  PRINT#3,"unter dem Namen ";la$;" gespeichert."
3490  FOR t=1 TO 2000:NEXT
3500  RETURN
3510  SOUND 1,200,100,7:LOCATE#3,13,1
3520  PRINT#3,"Noch keine Rechnung im Speicher..."
3530  FOR t=1 TO 2000:NEXT:RETURN
3540  REM***Unterprogramm NAME ZU LANG***
3550  SOUND 1,100,100,7:LOCATE#3,13,1
3560  PRINT#3"Der eingegebene Name ist zu lang"
3570  RETURN
```

Programmbeschreibung zum Baustein 4 - Rechnung speichern

Zeile 3000: Modus zuweisen
Zeile 3010: Bildschirm löschen und Modus wählen
Zeile 3020-3060: Alle WINDOWS erstellen
Zeile 3070: Text in WINDOW 3 setzen
Zeile 3080: Aktuellen Modus in WINDOW 1 ausgeben
Zeile 3090: Kommentarzeile
Zeile 3100: Ist die Variable MAXIMAL=0, dann ist keine Rechnung im Speicher
Zeile 3110: Text in WINDOW 2 ausgeben
Zeile 3120: Text in WINDOW 2 ausgeben
Zeile 3130: Tastaturabfrage
Zeile 3140: Text aus WINDOW 2 wieder löschen
Zeile 3150: Textausgabe in WINDOW 2
Zeile 3160: Textausgabe in WINDOW 2
Zeile 3170: Textcursor positionieren
Zeile 3180: Inhaltsverzeichnis zeigen
Zeile 3190: Textausgabe in WINDOW 3
Zeile 3200: Textausgabe in WINDOW 0
Zeile 3210: Eingabe, unter welchem Namen die Rechnung gespeichert werden soll
Zeile 3220: Abfrage, ob der Name länger als 8 Zeichen lang ist
Zeile 3230: Datei mit dem jeweiligen Namen öffnen
Zeile 3240-3450: Alle wichtigen Daten in die Datei schreiben
Zeile 3460: Datei wieder schließen
Zeile 3470: Textausgabe in WINDOW 3
Zeile 3480: Textausgabe in WINDOW 3
Zeile 3490: Warteschleife
Zeile 3500: Rücksprung ins Hauptmenü
Zeile 3510: Warnsignal ausgeben und Cursor positionieren
Zeile 3520: Textausgabe in WINDOW 3
Zeile 3530: Warteschleife und Rücksprung
Zeile 3540: Kommentarzeile
Zeile 3550: Warnsignal ausgeben und Cursor positionieren
ZEile 3560: Textausgabe in WINDOW 3
Zeile 3570: Rücksprung aus dem Unterprogramm

Programmlisting Baustein 5

```
4000 modus$="Modus: Bidschirmausgabe"
4010 MODE 2:CLS:i$=CHR$(24)
4020 WINDOW#1,1,80,1,2:PAPER#1,1:CLS#1:PEN#1,0
4030 WINDOW#2,1,80,4,23:PAPER#2,1:CLS#2:PEN#2,1
4040 WINDOW#3,1,80,25,25:PAPER#3,1:CLS#3
```

```
4050  PEN#3,0:PRINT#3,"Meldungen:";
4060  PEN#1,0:LOCATE#1,2,1:PRINT#1,modus$
4070  LOCATE#1,2,2:PRINT#1,"Nr.Art.-Nr.";
4080  PRINT#1,"Artikel-Bezeichnung         DM/Stck.";
4090  PRINT#1,"   Menge       Gesamtsumme"
4100  REM******Ausgabe auf Bildschirm******
4110  z=0
4120  FOR i=1 TO maximal
4130  z=z+1:IF artnr$(i)="ende" OR artnr$(i)="Ende" THEN 4250
4140  IF z=8 THEN GOSUB 4450
4150  PRINT#2,USING "###";i;
4160  PRINT#2,TAB(8);USING"\        \";artnr$(i);
4170  PRINT#2,TAB(19);USING"\                       \";artbez$(i);
4180  PRINT#2,TAB(44);USING"####.##";ep(I);
4190  menge$(i)=STR$(menge(i))
4200  PRINT#2,TAB(57);USING"\       \";menge$(I);
4210  PRINT#2,TAB(69);USING"#######.##";summe(i);
4220  PRINT#2:PRINT#2:
4230  NEXT
4240  REM*******Schlußrechnung******
4250  PRINT#2,STRING$(80,154);
4260  PRINT#2,TAB(69);USING"#######.##";gesamtsumme;
4270  PRINT#2:PRINT#2:PRINT#2
4280  PRINT#2,"Mehrwersteuerbetraege:":PRINT#2
4290  PRINT#2,"zum Satz von 7%      ";
4300  PRINT#2,"zum Satz von 14%     ";
4310  PRINT#2,"Insgesamt in DM"
4320  FOR d=1 TO 70:PRINT#2,"-";:NEXT d
4330  PRINT#2,"      ";mwst1;
4340  PRINT#2," DM",mwst2;" DM",
4350  PRINT#2,"      ",mwges;" DM "
4360  PRINT#2
4370  PRINT#2,TAB(46);i$;" Gesamtbetrag: ";endbetrag;" DM ";i$
4380  CLS#3:PRINT#3," Meldung: ";
4390  PRINT#3,"Keine weiteren Artikel mehr vorhanden.";
4400  PRINT#3," Taste druecken....."
4410  e$=INKEY$:IF e$="" THEN 4410 ELSE GOTO 4440
4420  bl$=INKEY$:IF bl$="" THEN 4420
4430  RETURN
4440  RETURN
4450  PRINT#3," Bitte eine Taste druecken......."
4470  GOSUB 4420
4480  RETURN
```

Programmbeschreibung zum Baustein 5 - Bildschirmausgabe

Zeile 4000: Modus zuweisen
Zeile 4010: Bildschirm löschen und Modus wählen
Zeile 4020-4060: Alle WINDOWs erstellen
Zeile 4070-4090: Tabellenüberschrift ausgeben
Zeile 4100: Kommentarzeile
Zeile 4110: Zähler (Z) auf Null setzen
Zeile 4120: Schleife für Ausgabe öffnen

Zeile 4130: Zähler um 1 erhöhen. Abfrage, ob das Ende schon erreicht wurde
Zeile 4140: Abfrage, ob der 8. Artikel schon erreicht wurde
Zeile 4150-4220: Ausgabe aller Artikel komplett
Zeile 4230: Schleife wieder schließen
Zeile 4240-4400: Schlußausgabe und Gesamtüberblick
Zeile 4410: Tastaturabfrage
Zeile 4420-4430: Unterprogramm Tastaturabfrage
Zeile 4440: Rücksprung aus diesem Programmteil
Zeile 4450-4470: Unterprogramm Meldung ausgeben

Programmlisting Baustein 6

```
5000 modus$="Modus: Kundenanschrift laden / eingeben"
5010 MODE 2:CLS:i$=CHR$(24)
5020 WINDOW#1,1,80,1,2:PAPER#1,1:CLS#1:PEN#1,0
5030 WINDOW#2,1,80,4,23:PAPER#2,0:CLS#2:PEN#2,1
5040 WINDOW#3,1,80,25,25:PAPER#3,1:CLS#3
5050 PEN#3,0:PRINT#3,"Meldungen:";
5060 PEN#1,0:LOCATE#1,2,1:PRINT#1,modus$
5070 LOCATE#2,1,2:PRINT#2,"Bitte wählen Sie: ":PRINT#2
5080 PRINT#2,"1 = Kundenanschrift aus Datei laden"
5090 PRINT#2
5100 PRINT#2,"2 = Kundenanschrift manuell eingeben "
5110 PRINT#2
5120 a$=INKEY$:IF a$="" THEN 5120
5130 IF a$="1" THEN 5600
5140 IF a$="2" THEN 5170
5150 PRINT#3,"-------------------------"
5160 SOUND 1,200,50,7:SOUND 2,201,50,7:GOTO 5120
5170 CLS#2:LOCATE#2,1,2
5180 INPUT#2,"Bitte Rechnungsdatum eingeben : ",date$
5190 INPUT#2,"Bitte Rechnungsnr.: ",rnr$
5200 PRINT#2
5210 INPUT#2,"Anrede:                    ",anrede$
5220 PRINT#2
5230 INPUT#2,"Firmenname  1. Zeile:      ",name1$
5240 PRINT#2
5250 INPUT#2,"Firmenname  2. Zeile:      ",name2$
5260 PRINT#2
5270 INPUT#2,"Strasse oder Postfach:     ",strasse$
5280 PRINT#2
5290 INPUT#2,"PLZ und Ort:               ",ort$
5300 PRINT#2
5310 INPUT#2,"Land:                      ",land$
5320 PRINT#2
5330 CLS#2
5340 PRINT#2,TAB(10);"Ihre eingegebene ";
5350 PRINT#2,"Anschrift sieht nun so aus: "
5360 PRINT#2
5370 PRINT#2
5380 PRINT#2,TAB(20);anrede$
5390 PRINT#2
```

```
5400  PRINT#2,TAB(20);name1$
5410  IF name2$="" THEN GOTO 5440
5420  PRINT#2
5430  PRINT#2,TAB(20);name2$
5440  PRINT#2
5450  PRINT#2,TAB(20);strasse$
5460  PRINT#2
5470  PRINT#2,TAB(20);ort$
5480  l=LEN(ort$)
5490  PRINT#2,TAB(20);:FOR i=1 TO l:PRINT#2,"=";:NEXT
5500  PRINT#2:PRINT#2
5510  IF land$="" THEN 5530
5520  PRINT#2,TAB(20);land$
5530  PRINT#2:PRINT#2:PRINT#2,STRING$(79,154)
5540  PRINT#2,"Anschrift so richtig eingegeben ?  ";
5550  PRINT#2,"j = ja  n = nein "
5560  e$=INKEY$: IF e$="" THEN 5560
5570  IF e$="j" THEN 5800
5580  IF e$="n" THEN 5170
5590  SOUND 1,300,50,7:SOUND 2,301,50,7:GOTO 5560
5600  CLS#2
5610  PRINT#2,"Bitte Cassette oder Diskette vorbereiten."
5620  PRINT#2,"Wenn Sie fertig sind, ";
5630  PRINT#2,"drücken Sie eine Taste...."
5640  e$=INKEY§: IF e$="" THEN 5640
5650  PRINT#2,"Bitte geben Sie die Kundennumer für die "
5660  INPUT#2,"gesuchte Anschrift des Kunden: ",kdnr$
5670  OPENIN"kunden"
5680  FOR i=1 TO 200
5690  IF EOF THEN 5790
5700  INPUT#9,kdnr$(i)
5710  INPUT#9,anrede$(i)
5720  INPUT#9,name$(i)
5730  INPUT#9,name2$(i)
5740  INPUT#9,strasse$(i)
5750  INPUT#9,ort$(i)
5760  INPUT#9,land$(i)
5770  IF kdnr$(i)=kdnr$ THEN GOSUB 5910:GOTO 5790
5780  NEXT
5790  CLOSEIN
5800  CLS#2:PRINT#2,"Wollen sie einen Werbetext erstellen?"
5810  PRINT#2:PRINT#2,"j = ja  n = nein"
5820  e$=INKEY$: IF e$="" THEN 5820
5830  IF e$="n" THEN 5890
5840  LOCATE#2,1,7:PRINT#2,"Textzeile 1: ";:LINE INPUT#2,"",text1$
5850  LOCATE#2,1,8:PRINT#2,"Textzeile 2: ";:LINE INPUT#2,"",text2$
5860  LOCATE#2,1,10:PRINT#2,"so richtig?
      ":PRINT#2:PRINT#2,text1$:PRINT#2,text2$
5870  e$=INKEY$:IF e$="" THEN 5870
5880  IF e$="n" THEN 5800
5890  REM **** zurück ins Hauptmenü ****
5900  RETURN
5910  REM *** Anschrift gefunden ***
5920  SOUND 1,100,50,2
5930  PRINT#2,"Anschrift gefunden"
5940  RETURN
```

Programmbeschreibung zum Baustein 6 - Kundenanschrift laden

Zeile 5000: Modus zuweisen
Zeile 5010: Bildschirm löschen und Modus wählen
Zeile 5020-5060: Alle WINDOWs erstellen
Zeile 5070: Textausgabe
Zeile 5080-5110: Hilfsmenü anbieten
Zeile 5120: Tastaturabfrage
Zeile 5130-5140: Abfrage, was gedruckt wurde
Zeile 5150-5160: Bei falschen Eingaben wieder zurück
Zeile 5170: Bildschirm löschen und Cursor positionieren
Zeile 5180-5190: Datum und Rechn.-Nr. eingeben
Zeile 5200-5540: Kunden komplett erfassen
Zeile 5550-5590: Kunden richtig eingegeben ?
Zeile 5600-5790: Kunden über die Kundennummer aus der Datei laden
Zeile 5800-5890: Werbetext erstellen - max. 2 Zeilen
Zeile 5900: Rücksprung ins Hauptmenü mit RETURN
Zeile 5910-5940: Hinweis ausgeben, daß Kunde gefunden wurde

Programmlisting Baustein 7

```
6000 modus$="Modus: Druckerausgabe "
6010 MODE 2:CLS:i$=CHR$(24)
6020 br$=CHR$(&E)
6030 WINDOW#1,1,80,1,2:PAPER#1,1:CLS#1:PEN#1,0
6040 WINDOW#2,1,80,4,23:PAPER#2,0:CLS#2:PEN#2,1
6050 WINDOW#3,1,80,25,25:PAPER#3,1:CLS#3
6060 PEN#3,0:PRINT#3,"Meldungen:";
6070 PEN#1,0:LOCATE#1,2,1:PRINT#1,modus$
6080 REM ******* Ausgabe auf Drucker *******
6090 PRINT#8,CHR$(27);"§"
6100 REM *******Firmenkopf des Absenders *******
6110 PRINT#8,br$;"**************************************"
6120 PRINT#8
6130 PRINT#8,br$;"C B V - COMPUTER BERATUNG UND VERTRIEB"
6140 PRINT#8
6150 PRINT#8,TAB(10);"Berliner Str. 14  4830 Guetersloh 1";
6160 PRINT#8,SPC(6);"Tel. 05241-15186":PRINT#8
6170 PRINT#8,br$;"****************************************************"
6180 REM *******Ende des Firmenkopfs **********
6190 REM
6200 REM
6210 FOR abstand=1 TO 4:PRINT#8:NEXT abstand
6220 REM *****          Empfängeranschrift         ******
6230 REM ********* für das Sichtfenster im Brief ******
6240 x=8: REM  Tabulatorabstand von links
6250 PRINT#8,TAB(x);anrede$
6260 PRINT#8,TAB(x);name1$
6270 IF name2$<>"" THEN PRINT#8,TAB(x);name2$
6280 PRINT#8,TAB(x);strasse$
```

```
6290  PRINT#8
6300  PRINT#8,TAB(x);ort$
6310  FOR i = x TO LEN(ort$)+x
6320  PRINT#8,TAB(i);"=";
6330  NEXT i
6340  PRINT#8:PRINT#8
6350  IF land$ <> "" THEN PRINT#8,TAB(x);br$;land$
6360  REM ******** Anschrift ins Fenster geschrieben ******
6370  REM
6380  REM
6390  PRINT#8,TAB(50);"Rechnungsdatum: ";date$
6400  PRINT#8,TAB(50);"Rechnungsnr.:  ";rnr$
6410  PRINT#8
6420  PRINT#8,TAB(15);br$;"R E C H N U N G"
6430  FOR i=1 TO 4:PRINT#8:NEXT
6440  PRINT#8,"Nr.  Art.-Nr.  ";
6450  PRINT#8,"Artikel-Bezeichnung  DM/Stck.";
6460  PRINT#8,"  Menge  Gesamtsumme"
6470  PRINT#8,STRING$(80,"=")
6480  FOR i=1 TO maximal
6490  IF artnr$(i)="ende" OR artnr$(i)="Ende" THEN 6600
6500  PRINT#8,USING "###";i;
6510  PRINT#8,TAB(7);USING"\       \";artnr$(i);
6520  PRINT#8,TAB (17);USING"\                        \";artbez$(i);
6530  PRINT#8,TAB(44);USING"####.##";ep(I);
6540  menge$(i)=STR$(menge(i))
6550  PRINT#8,TAB(54);USING"\     ";menge$(I);
6560  PRINT#8,TAB(66);USING"#######.##";summe(i);
6570  PRINT#8:PRINT#2:
6580  PRINT#8:NEXT
6590  REM ******Schlussrechnung ******
6600  PRINT#8,STRING$(80,"=");
6610  PRINT#8,TAB(66);USING"#######.##";gesamtsumme;
6620  PRINT#8:PRINT#8:PRINT#8
6630  PRINT#8,"Mehrwertsteuerbeträge:"
6640  PRINT#8,"zum Satz von 7 %      ";
6650  PRINT#8,"zum Satz von 14 %     ";
6660  PRINT#8,"Insgesamt in DM"
6670  FOR d=1 TO 65:PRINT#8,"-";:NEXT d:PRINT#8
6680  PRINT#8,"         ";mwst1;
6690  PRINT#8," DM",mwst2;" DM",
6700  PRINT#8," ",mwges;" DM "
6710  PRINT#8
6720  PRINT#8,TAB(44);"Gesamtgetrag in DM:";
6730  PRINT#8,TAB(66);USING"#######.##";endbetrag
6740  PRINT#8
6750  PRINT#8,"*** Zahlungsbedingungen: Innerhalb 10 Tagen";
6760  PRINT#8," 2% Skonto, 30 Tage netto Kasse."
6770  PRINT#8
6780  PRINT#8,text1$
6790  PRINT#8,text2$
6800  PRINT#8,CHR$(27);"§"
6810  CLS#3:PRINT#3," Meldung: ";
6820  PRINT#3,"Keine weiteren Artikel mehr vorhanden.";
6830  PRINT#3," Taste druecken...."
6840  e$=INKEY$:IF e$="" THEN 6840 ELSE GOTO 6850
6850  RETURN
```

Programmbeschreibung zum Baustein 7 - Druckerausgabe

Einer der umfangreichsten Programmteile in dem Programm zur Rechnungsschreibung ist der Baustein zum Drucken der Rechnung. Vorweg noch ein paar Worte zum Drucker. Ich habe bei allen Listings einen preiswerten Matrixdrucker mit Centronics-Schnittstelle verwendet, einen CP 80. Im Programm erscheinen einige Zeichen zur Ansteuerung für diesen Drucker, z.B. um die Breitschrift einzuschalten oder den Drucker zu initialisieren. Wenn Sie einen anderen Drucker verwenden, müssen diese Steuercodes einfach angepaßt werden. In Ihrem Druckerhandbuch finden Sie die entsprechenden Angaben.

In diesem Baustein brauchen Sie aber nur zwei Änderungen vorzunehmen, wenn Sie einen anderen Drucker verwenden. In der Zeile 6020 finden Sie Variable BR$. Dieser Variablen wird der Code für die Breitschrift zugeordnet. Bei Verwendung eines anderen Druckers sollten Sie in Ihrem Druckerhandbuch nachschlagen und den entsprechenden Code für die Breitschrift der Variablen BR$ zuordnen.

Die zweite Änderung betrifft die Zeilen 6090 und 6800. In diesen Zeilen wird der Drucker initialisiert, was bedeutet, daß der Drucker in seinen Einschaltzustand versetzt wird und seine Standardwerte annimmt. Viele Drucker verwenden für die Initialisierung die Codefolge

```
PRINT chr$(27);"@";
```

oder, wenn kein Klammeraffe vorhanden ist, die Folge

```
PRINT CHR$(27);CHR$(64)
```

In Ihrem Handbuch finden Sie auch dazu weitere Hinweise, wie Ihr Druckermodell initialisiert wird.

Nun zum eigentlichen Programm. Es druckt als erstes einen Firmenkopf, dann die Anschrift des Kunden, das Rechnungsdatum und die Rechnungsnummer. Anschließend werden alle Artikel mit Mengen und Preisen ausgedruckt.

Der Programmteil Rechnung gibt später eine Rechnung mit detaillierten Angaben über die Mehrwertsteuer und den Rechnungsbetrag aus. Darunter erscheinen noch die Zahlungsbedingungen und zwei Zeilen mit einem beliebigen Werbetext.

Zeile 6000: Modus festlegen
Zeile 6010: Bildschirm einstellen

Zeile 6020: Breitschrift der Variablen BR$ zuweisen
Zeile 6030-6050: WINDOW 1 bis 3 definieren und Farbe setzen
Zeile 6060: Textausgabe in WINDOW 3
Zeile 6070: Modus in WINDOW 1 ausgeben
Zeile 6080: Kommentarzeile
Zeile 6090: Drucker intialisieren
Zeile 6100: Kommentarzeile
Zeile 6110: In Breitschrift eine Reihe von Sternen ausgeben
Zeile 6120: Leerzeile auf dem Drucker
Zeile 6130: In Breitschrift den Firmennamen ausgeben
Zeile 6140: Leerzeile auf dem Drucker
Zeile 6150-6160: Anschrift ausgeben und weitere Leerzeile einfügen
Zeile 6170: Sterne in Breitschrift ausgeben, wie in Zeile 6110
Zeile 6180-6200: Kommentarzeile
Zeile 6210: 4 Zeilen Abstand erzeugen zwischen dem Firmenkopf und der folgenden Anschrift
Zeile 6220-6230: Kommentarzeile
Zeile 6240: Tabulator auf den Wert 8 einstellen (wird benötigt für das Sichtfenster bei Briefumschlägen)
Zeile 6250-6260: Anrede und Name ausdrucken
Zeile 6270: Wenn ein zweiter Name eingegeben wurde, dann wird dieser auch ausgedruckt
Zeile 6280-6290: Straße und Hausnummer oder das Postfach ausdrucken
Zeile 6300: Ort mit Postleitzahl ausdrucken
Zeile 6310-6330: Ort und Postleitzahl auf der ganzen Länge mit einem Zeichen (=) unterstreichen
Zeile 6340: Zwei Leerzeilen lassen zwischen dem Ort und dem nachfolgendem Ausdruck
Zeile 6350: Wenn als Land etwas eingegeben wurde, dann wird dies in Breitschrift ausgegeben
Zeile 6360-6380: Kommentarzeile
Zeile 6390-6400: Rechnungsdatum und Rechnungsnummer ausdrucken
Zeile 6410: Leerzeile ausgeben
Zeile 6420: An der Position 15 das Wort "Rechnung" in Breitschrift ausdrucken
Zeile 6430: Vier Leerzeilen ausgeben
Zeile 6440-6470: Lfd. Nr., Artikel-Nr., Artikel-Bezeichnung, Einzelpreis, Menge und Gesamtsumme ausdrucken
Zeile 6480: Schleife zur Ausgabe der einzelnen Rechnungspositionen öffnen

Zeile 6490: Wenn bei der Artikel-Nr. "ende" eingegeben wurde, dann zur Zeile 6600 gehen
Zeile 6500-6570: Die einzelnen Positionen der Rechnung formatiert ausdrucken
Zeile 6580: Leerzeile drucken und Schleife schließen
Zeile 6590: Kommentarzeile
Zeile 6600: Linie aus 80 Gleichheitszeichen drucken
Zeile 6610: Nettogesamtbetrag formatiert ausdrucken
Zeile 6620: Drei Leerzeilen ausgeben
Zeile 6630-6700: Mehrwertsteuerbeträge formatiert ausdrucken, getrennt nach 7 % und 14 % Mehrwertsteuer
Zeile 6710: Leerzeile ausgeben
Zeile 6720-6730: Bruttobetrag ausdrucken (incl. Mehrwertsteuer)
Zeile 6740: Leerzeile ausgeben
Zeile 6750-6760: Zahlungsbedingungen ausdrucken
Zeile 6770: Leerzeile ausgeben
Zeile 6780: Erste Zeile für Werbetext drucken
Zeile 6790: Zweite Zeile für Werbetext drucken
Zeile 6800: Drucker intialisieren (Druckerpuffer leeren)
Zeile 6810-6830: Meldung in WINDOW 3 ausgeben
Zeile 6840: Abfrage der Tastatur
Zeile 6850: Rücksprung ins Hauptmenü mit RETURN

Programmlisting Baustein 8

```
7000 modus$="Modus: Programm beenden "
7010 MODE 2:CLS:i$=CHR$(24)
7020 WINDOW#1,1,80,1,2:PAPER#1,1:CLS#1:PEN#1,0
7030 WINDOW SWAP 0,2
7040 WINDOW#0,1,80,4,23:PAPER#0,0:CLS#0:PEN#0,1
7050 WINDOW#2,1,80,4,23:PAPER#2,0:CLS#2:PEN#2,1
7060 WINDOW#3,1,80,24,25:PAPER#3,1:CLS#3
7070 PEN#3,0:PRINT#3," Meldungen:";
7080 PRINT#3," Achtung! Daten und Programm werden ";
7090 PRINT#3,TAB(14);"bei Eingabe von j vernichtet !!"
7100 PEN#1,0:LOCATE#1,2,1:PRINT#1,modus$
7110 LOCATE#2,1,2:PRINT#2,"Sind Sie ganz sicher, ";
7120 PRINT#2,"dass Sie Ihr Programm beenden wollen ?"
7130 PRINT#2
7140 PRINT#2," j = ja  n = nein"
7150 a$=INKEY$:IF a$="" THEN 7150
7160 IF a$="j" THEN 7200
7170 IF a$="n" THEN 7220
7180 SOUND 1,100,100,7:SOUND 2,101,100,7
7190 GOTO 7150
7200 CALL (0)
7210 '****** zurück ins Hauptmenü **********
7220 RETURN
```

Programmbeschreibung für Baustein 8 - Programm beenden

Wenn Sie im Menü diesen Punkt anwählen, werden Sie noch gefragt, ob das Programm auch wirklich beendet werden soll. Drücken Sie die Taste j, dann löscht sich das Programm von selbst und Sie bekommen wieder die Systemmeldung des Schneiders auf den Bildschirm. Haben Sie versehentlich ein falsches Menue gewählt, dann können Sie auf die Taste n drücken und das Programm springt automatisch wieder zurück ins Hauptmenü.

Hier nun die Beschreibung der einzelnen Programmzeilen:

Zeile 7000: Modus wählen
Zeile 7010: Bildschirm einstellen
Zeile 7020: WINDOW 1 definieren und Farben setzen
Zeile 7030: WINDOW 0 und 2 vertauschen (Systemmeldung)
Zeile 7040-7060: WINDOW 0, 2 und 3 definieren und Farben setzen
Zeile 7070: in WINDOW 3 einen Text setzen
Zeile 7080: in WINDOW 3 einen weiteren Text setzen
Zeile 7090: wie Zeile 7080
Zeile 7100: in WINDOW 1 den Modus angeben, in dem man sich gerade befindet
Zeile 7110-7140: Text in WINDOW 2 setzen
Zeile 7150: Abfrage der Tastatur
Zeile 7160: Wird die Taste j gedrückt dann zur Zeile 7200 gehen und das Programm beenden
Zeile 7170: Wenn die Taste n gedrückt wird, dann zur Zeile 7220 gehen. Hier geht es wieder zurück ins Hauptmenü.
Zeile 7180: Über den Kanal 1 und 2 zwei Töne als Warnsignal ausgeben für eine fehlerhafte Eingabe
Zeile 7190: Rücksprung zur Tastaturabfrage in Zeile 7150
Zeile 7200: Mit CALL (0) das gesamte System in den Einschaltzustand setzen. Dadurch werden das Programm und alle eingegebenen Daten vernichtet.
Zeile 7210: Kommentarzeile
Zeile 7220: Rücksprung ins Menü mit RETURN

```
****************************************

C B V  -  COMPUTER BERATUNG UND VERTRIEB

     Berliner Str. 14     4830 Guetersloh 1     Tel. 05241-15186

****************************************


     Fa.
     Computer-Store
     Inh. D. Buschkamp
     Schulstr. 9

     4830 Guetersloh 1
     ==================

                                   Rechnungsdatum: 25.5.1985
                                   Rechnungsnr.:   005busch

           R  E  C  H  N  U  N  G


Nr.  Art.-Nr.  Artikel-Bezeichnung      DM/Stck.  Menge     Gesamtsumme
=========================================================================
 1   edv01     BASIC fuer Einsteiger       29.66  3               88.98

 2   edv02     CPC 464-Handbuch            44.77  3              134.31

 3   edv03     EDV-Papier 2000 Bl.         29.77  12             357.24

 4   edv04     Druckerkabel                45.88  2               91.76

 5   edv05     3"-Disketten                 9.80  1000          9800.00

 6   edv06     5.25"-Disketten              4.88  10              48.80

 7   edv07     Drucker FX 100            1789.55  1             1789.55

 8   edv08     Farbbandcassette            12.66  4               50.64

 9   edv09     COMMODORE PC 10           4500.00  1             4500.00

=========================================================================
                                                               16861.28


Mehrwertsteuerbetraege:
zum Satz von 7 %      zum Satz von 14 %       Insgesamt in DM
------------------------------------------------------------------
        15.63  DM         2329.32  DM             2344.95  DM

                                   Gesamtbetrag in DM:      19206.23

*** Zahlungsbedingungen:  Innerhalb 10 Tage 2% Skonto, 30 Tage netto Kasse.

Beachten Sie auch unser Angebot an Akustikkopplern
und dazu passender Software.  Es lohnt sich !
```

Bild 20.2: *Eine ausgedruckte Rechnung*

21 Programmplanung

Wie Sie schon in den letzten Kapiteln gesehen haben, sollte man vor Beginn seiner Programmierarbeit seine Vorgehensweise festlegen, damit es später nicht zu unliebsamen Überraschungen kommt.

In diesem Kapitel geht es also nun um die reine Planung eines BASIC-Programms. Hier möchte ich Ihnen am Beispiel einer Mittelwertberechnung zeigen, wie man ein solches Programm plant.

Da man sich Abläufe am besten vorstellen kann, wenn man etwas sieht, wurden sogenannte Programmablaufpläne entwickelt. Dabei gibt es für alles, was ein Programm ausführen kann, ein bestimmtes Symbol. Diese Symbole sind nach DIN genormt und lassen sich recht vielseitig einsetzen. Mit Hilfe dieser Symbole setzt man sich nun auf dem Papier sein Programm zusammen und programmiert es später nach diesem Plan.

In welcher Programmiersprache dann dieses Programm erstellt wird, ist im Prinzip gleich, denn der logische Ablauf eines Programms ändert sich ja nicht.

Ist einmal ein solcher Programmablaufplan vorhanden, kann man ein Programm auf jeden anderen Computer übertragen, da die Logik des Programms ja bereits festliegt. Lediglich die einzelnen Befehle lauten später bei anderen Computersystemen etwas anders, arbeiten aber genauso.

Auf der folgenden Seite finden Sie eine Reihe von Symbolen, die gebräuchlich sind und mit deren Hilfe man Programme plant. Selbstverständlich ist dies nur ein Hilfsinstrument, um sich einen Ablauf besser vorzustellen.

Vielleicht entwickeln Sie später Ihre eigene Methode und kommen damit noch besser klar. Auf jeden Fall sollten Sie vor allem größere Projekte nach irgendeiner Methode planen, damit später nichts schief geht.

Symbol	Erklärungen / Bemerkungen / Beispiele
	Programmstart und Programmende
	Übergangsstelle zu einem anderen Programm Beispiel: Datenschnittstelle.
	Allgemeine Operationen, z. B. eine Berechnung
	Logischer Vergleich mit nur zwei Ausgängen Ausgang 1: Ja, Vergleich trifft zu Ausgang 2: Nein, Vergleich trifft nicht zu
	Unterprogramm, z. B. Tastaturabfrage
	Dateneingabe oder Datenausgabe Beispiele: Mit PRINT etwas ausgeben, mit INPUT etwas eingeben
	Datenflußlinien

Bild 21.1: *Die Symbole eines Programmablaufplans*

Nachdem Sie auf der vorherigen Seite die wichtigsten Symbole kennengelernt haben, möchte ich Ihnen nun ein einfaches Beispiel zeigen, wie man diese Symbole einsetzt.

Aufgabe:

Ein Programm zur Berechnung der Fläche eines Rechtecks soll geschrieben werden.

Eingabe:

Länge
Breite

Verarbeitung:

Fläche = Länge * Breite

Ausgabe:

Länge, Breite und Fläche

Der Programmablaufplan

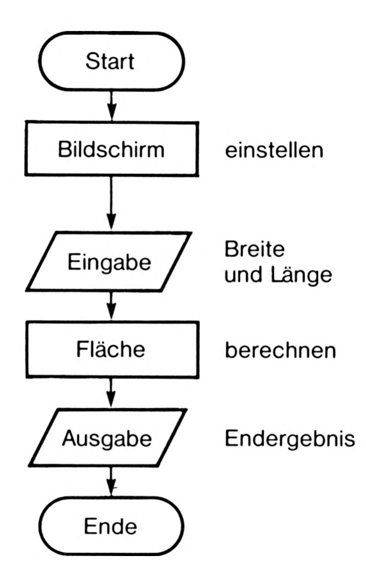


Bild 21.2

Das Programm

```
10    MODE 1:CLS
20    BORDER 0:INK 0,0
30    INPUT"LAENGE";L
40    INPUT"BREITE";B
50    FL=L*B
60    PRINT"LAENGE: ";L
70    PRINT"BREITE:";B
80    PRINT"FLAECHE:";FL
90    END
```

Wie man sieht, wird das Programm eigentlich in drei wichtige Teile eingeteilt:

a. Eingabe
b. Verarbeitung
c. Ausgabe

Neben der richtigen Kombination der einzelnen Symbole sollten Sie sich unbedingt eine Tabelle mit den im Programm verwendeten Variablen anlegen, damit eine Dopellbelegung später ausgeschlossen ist. Eine Tabelle dazu könnte beispielsweise so aussehen:

Variable	**wird verwendet als**
I	Laufvariable für Schleifen
PAUSE	Laufvariable für Warteschleife
ANZAHL	Zähler für Eingaben
SUMME	Laufende Summe
....	....
....	....

Auf dieses Hilfsmittel sollten man also nicht verzichten, selbst ein paar Notizen auf einem Schmierzettel erfüllen schon ihren Zweck. So erhält man einen schnellen Überblick beim Programmieren, welche Variablen man schon vergeben und welche noch zur Verfügung stehen.

Im letzen Kapitel wurde ebenfalls schon besprochen, daß man Unterprogramme einsetzen sollte, wo es möglich ist. Dadurch erhält man eine Struktur innerhalb eines Programms. Ich setze z.B. fast alle Unterprogramme immer an das Ende eines Programms und rufe sie dann vom Hauptprogramm auf.

Schematisch betrachtet sieht dies so aus:

```
    Programm initialisieren
```

```
H A U P T - P R O G R A M M

Unterprogramm 1

Unterprogramm 2

Unterprogramm ....
```

Eine sehr wichtige Rolle spielen beim Programmieren die einzelnen Zeilennummern. Beispielsweise ist man sich bei einem **GOTO**-Befehl am Anfang des Programms nicht sicher, wo er denn später ganz genau hingehen soll. Hier verfahre ich wie folgt. Angenommen man möchte mit dem Befehl **GOTO** zu irgendeiner Auswertung im hinteren Teil des Programms springen, der aber noch nicht fertig ist. In einem solchen Fall schreibe ich die Zeilennummer vorläufig so:

```
100 GOTO Auswertung
```

Wenn das Programm dann später fertig ist, wird diese Zeile entsprechend angepaßt.

Die Verwendung von **GOTO**-Befehlen ist also schon etwas problematisch, deshalb verwende ich auch lieber den Befehl **GOSUB**.

In diesem Zusammenhang möchte ich auch noch einmal die Unterprogramme ansprechen, speziell hier die Zeilennumerierung. Als Einsprungsadressen wähle ich möglichst gerade Zeilennummern, da man diese geraden Nummern beim Programmieren leichter behalten kann.

Auf der letzten Seite haben Sie in dem Schema im ersten Teil die Bezeichnung "initialisieren" gesehen. Damit ist folgendes gemeint: Bei vielen Programmen, die mit indizierten Variablen arbeiten, müssen vorher Felder dimensioniert werden. Außerdem bietet sich hier der Platz geradezu an für Bemerkungen, Copyright-Hinweise und Bezeichnungen der einzelnen Variablen, also für alles, was nur einmal durchlaufen werden soll.

Hier bieten sich vor allem die Zeilennummern von 1 bis 99 an.

Beispiel:

```
10 REM *********************************************
20 REM **   Flächenberechnung eines Rechtecks     **
30 REM **    (c) 1985 by Otto Meier               **
40 REM **   Version 1.0    Stand Juni 1985        **
50 REM *********************************************
60 DIM A(100),B(100)
```

Programmbeispiel: Mittelwertberechnung

Aufgabe:

Es soll aus einer beliebigen Anzahl von Werten der Durchschnitt ermittelt werden.

Vorgabe:

a. Eingabe von beliebig vielen Werten
b. Berechnung des Mittelwerts bei Eingabe von -999
c. Ausgabe der Zwischenwerte von Summe und Anzahl der Eingaben
d. Ausgabe des Mittelwerts, der Gesamtsumme der Eingaben und der Anzahl aller gemachten Eingaben bis zur Endberechnung

Vorarbeit:

Programmablaufplan erstellen

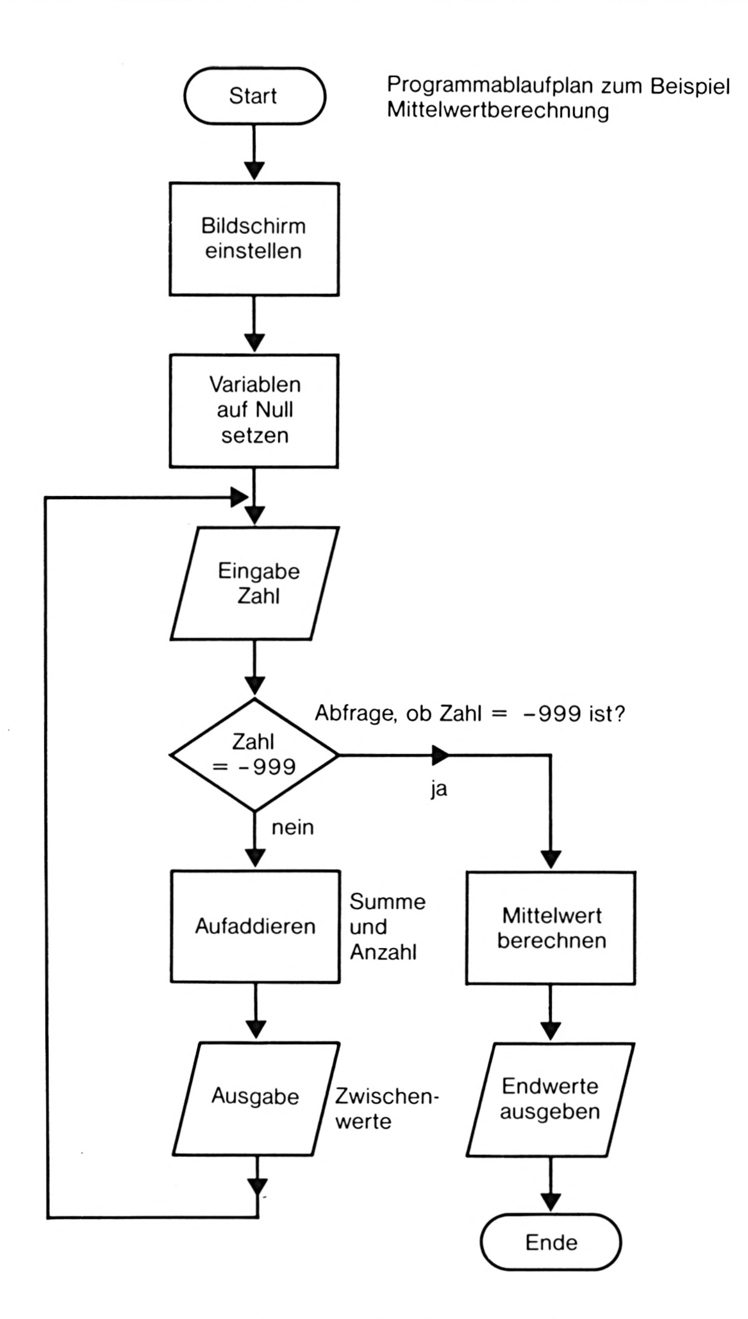


Bild 21.3: *Programmablaufplan zur Mittelwertberechnung*

```
10    REM*****************************************
20    REM**  Aufgabe: Mittelwerberechnung       **
30    REM**  Variablen: SUMME                   **
40    REM**             ANZAHL                  **
50    REM**             ZAHL                    **
60    REM**             MITTELWERT              **
70    REM*****************************************
80    REM
90    REM
100   MODE 1
110   CLS
120   INK 0,0
130   BORDER 0
140   REM **** Verwendete Variablen auf Null setzen ****
150   SUMME=0
160   ANZAHL=0
170   ZAHL=0
180   REM **** Dateneingabe ****
190   INPUT"Bitte geben Sie eine Zahl ein  ";ZAHL
200   REM **** Abfrage, Abbruchbedingung erfüllt? ****
210   IF ZAHL= -999 THEN GOTO 360
220   REM **** Summe aufaddieren ****
230   SUMME=SUMME + ZAHL
240   REM **** Anzahl der Eingaben zählen ****
250   ANZAHL=ANZAHL + 1
260   REM **** Ausgabe der aktuellen Werte ****
270   CLS
280   PRINT"Sie haben eben eingegeben: ";ZAHL
290   PRINT"Das war die ";ANZAHL;".te Eingabe"
300   PRINT"Die laufende Summe beträgt jetzt: "SUMME
310   PRINT:PRINT:PRINT"Weiter mit einer Taste.....
320   REM **** Abfrage der Tastatur ****
330   e$=INKEY$: IF e$="" THEN 330
340   REM **** Rücksprung zur nächsten Eingabe ***
350   CLS:GOTO 190
360   REM **** Berechnung des Mittelwerts ****
370   MITTELWERT=SUMME / ANZAHL
380   REM **** Ausgabe aller Daten auf dem Bildschirm ****
390   CLS:PRINT"    - Endberechnung - "
400   PRINT
410   PRINT"Anzahl der Eingaben: ";TAB(25);ANZAHL
420   PRINT"Gesamtsumme:         ";TAB(25);SUMME
430   PRINT"Mittelwert:          ";TAB(25);MITTELWERT
```

Programmbeschreibung:

Zeile 10 bis 90: Kommentarzeilen zur Dokumentation
Zeile 100 bis 130: Bildschirm einstellen
Zeile 140: Kommentarzeile
Zeile 150 bis 170: Verwendete Variablen auf Null setzen
Zeile 180: Kommentarzeile
Zeile 190: Eingabe einer Zahl
Zeile 200: Kommentarzeile

Zeile 210:	Wenn bei der Eingabe der Wert -999 eingegeben wird, dann die Endberechnung aufrufen
Zeile 220:	Kommentarzeile
Zeile 230:	Eingegebene Zahlen aufaddieren
Zeile 240:	Kommentarzeile
Zeile 250:	Anzahl der Eingaben aufaddieren
Zeile 260:	Kommentarzeile
Zeile 270:	Bildschirm löschen
Zeile 280:	Ausgabe der eingegebenen Zahlen
Zeile 290:	Ausgabe, um die wievielte Eingabe es sich handelt
Zeile 300:	Ausgabe der laufenden Summe
Zeile 310:	Leerzeilen und Textausgabe
Zeile 320:	Kommentarzeile
Zeile 330:	Abfrage der Tastatur
Zeile 340:	Kommentarzeile
Zeile 350:	Bildschirm löschen und Rücksprung
Zeile 360:	Kommentarzeile
Zeile 370:	Mittelwert berechnen
Zeile 380:	Kommentarzeile
Zeile 390:	Bildschirm löschen und Textausgabe
Zeile 400:	Leerzeile ausgeben
Zeile 410:	Ausgabe der gemachten Eingaben
Zeile 420:	Ausgabe der Gesamtsumme
Zeile 430:	Ausgabe des Mittelwerts

Wie Sie also gesehen haben, ist es doch recht hilfreich, wenn man sich vor dem Programmieren ein paar Gedanken zum Ablauf des Programms macht. Zusammenhänge lassen sich so recht schnell erkennen und solche Programme sind später auch wesentlich änderungsfreundlicher, falls dies einmal notwendig werden sollte.

22 Fehlersuche in BASIC-Programmen

Ich habe beim Programmieren folgende Erfahrungen gemacht: Auf Anhieb läuft am Anfang nicht immer alles so, wie man es selbst gerne hätte. Zwischen der Idee zum Programm und dem fertigen Programm-Listing liegen oft ungeahnte Schwierigkeiten und Fehlerquellen, um die man sich am Anfang überhaupt nicht gekümmert hat. Hier gibt es im Prinzip nur eine Antwort:

Übung macht den Meister

Nur wer dauernd am Ball bleibt und die entsprechende Praxis im Programmieren hat, erkennt Fehler auf Anhieb oder zumindest auf den zweiten Blick. Da Sie als Einsteiger wahrscheinlich noch nicht über diese Erfahrungen verfügen, erhalten Sie in diesem Kapitel einige Tips und Tricks, um Fehler in erstellten Programmen auszumerzen.

Dabei sollten Sie sich auch das Kapitel 18 ansehen, in dem die einzelnen Fehlermeldungen im Detail erläutert werden. Sehr sinnvoll ist auch die Fehlerroutine im Kapitel 18, die Ihnen die Fehlermeldungen in Deutsch ausgibt.

Auf jeden Fall sollten Sie in den ersten Zeilennummern Ihres Programms zur Fehlerlokalisierung die folgenden Zeilen eingegeben werden:

```
1     ON ERROR GOTO 63500
63500 MODE 1
63510 CLS
63520 PRINT"Fehler in Zeile ";ERL
63530 PRINT"Fehlercode:     ";ERR
63540 END
```

Bei einem auftretenden Fehler verzweigt das Programm zur Zeile 63500, wo in den folgenden Zeilen der Fehler lokalisiert wird.

Der wohl häufigste Fehler beim Programmieren ist der schon bekannte Syntax-Error, der vielfältige Ursachen haben kann. Meistens hat man einen Abstand zwischen den einzelnen Befehlen vergessen, eine Variable grammatisch nicht richtig geschreiben oder einen Befehl innerhalb der angegebenen Zeile falsch eingetippt. Dieser Fehler ist auch gleichzeitig der angenehmste, weil man sich wahrscheinlich nur verschrieben hat. Die

Zeile, in der der Fehler steckt, ist schnell korrigiert und schon kann es im Programm weitergehen.

Eine fast genauso häufige Fehlerquelle sind Sprungbefehle, die auf eine Zeilennummer zeigen, die es im Programm nicht gibt. Diese Fehler treten bei der Verwendung von GOTO und GOSUB auf. Auch dieser Fehler wird gemeldet und man kann die fehlenden Zeilennummern nachträglich einfügen.

Beim Programmieren von Schleifen mit FOR...NEXT kann es geschehen, daß die Schleife nur ein einziges Mal durchlaufen wird. In diesem Fall fehlt meistens der Befehl NEXT zum Schließen der einmal geöffneten Schleife. Desweiteren muß man bei Schleifen aufpassen, daß man keine Endlosschleifen programmiert. Endlosschleifen sind Schleifen, die einen Programmierfehler haben und von sich aus die Schleife nicht wieder verlassen können. Meistens sieht man auf dem Bildschirm dann nichts. Hier hilft dann nur der Druck auf die Taste <ESC>, um aus dem laufenden Programm zu kommen.

Bei Unterprogrammen, die mit GOSUB aufgerufen werden, sollte man darauf achten, daß das Unterprogramm nur einen einzigen Ein- und Ausgang hat. Außerdem sollte ein Unterprogramm nur wieder mit RETURN verlassen werden und nicht etwa mit dem Befehl GOTO.

Oft treten Fehler bei der Verarbeitung von Variablen auf, mit denen man vorher nicht gerechnet hat.

Beispiel: Ein Produktivitätskennzahl soll errechnet werden.

Die Formel dazu lautet:

Kennzahl = gefertigte Stück / Vorgabe * 100 %

```
10    INPUT ISTSTUECK
20    INPUT SOLLSTUECK
30    KENNZAHL=ISTSTÜCK / SOLLSTUECK * 100 %
40    PRINT KENNZAHL
```

Wenn Sie das Programm starten und die beiden Werte für die gefertigten Stück und die Vorgabestück eingeben, dann erhalten Sie irgendeine Kennziffer. Sollten Sie aber bei der zweiten Frage (Vorgabestück) keine Daten eingeben, sondern nur die Taste <ENTER> betätigen, sieht das Endergebnis dann schon etwas anders aus. Sie haben also dann versucht, in Zeile 30 durch Null zu teilen, was nach den mathematischen Regeln nicht erlaubt ist. Achten Sie also vor wichtigen Berechnungen auf den Inhalt der Variablen.

Bei indizierten Variablen sollte man sich vorher vergewissern, ob diese auch ausreichend groß dimensioniert wurden. Ansonsten kommt es sehr schnell zu einer Fehlermeldung. Achten Sie zusätzlich daruaf, daß die DIM-Befehle möglichst am Anfang Ihres Programms stehen und nur einmal durchlaufen werden. Soweit nun zu den häufigsten Fehlerquellen.

Im folgenden Beispiel geht es um die Suche von Fehlern, die ich in das Programm eingebaut habe. Das Demoprogramm führt einige Berechnungen durch und gibt diese auf dem Bildschirm aus.

Das Programm:

```
10    MODE 3:INK 0,0:BORDER 0
20    CLS
30    FOR I=1 TO 15
40    A(I)=I*I
50    NEXT I
60    PRINT"Ende der ersten Berechnung"
70    DATA 1,2,3,4,5,6,7
80    FOR LESEN =1 TO 9
90    READ ZAHL
100   PRINT ZAHL
110   NEXT ZAHL
120   FOR PAUSE=1 TO 1000:NEXT I:GOTO 15
```

Vergessen Sie nicht die Zeilennummern für die Fehlerroutine, denn die wird Ihnen noch recht nützlich sein.

Bei der Fehlersuche in Programmen sollte man erstens die BASIC-Befehle kennen und zweitens auch wissen, wie diese Befehle zusammen wirken. Betrachten wir uns noch einmal die ersten beiden Zeilennummern. Sie haben die Aufgabe, den Bildschirm entsprechend anzupassen. Nach dem Start mit RUN erhalten Sie einen Fehler in der Zeile 10. Also sehen wir uns Befehl für Befehl in dieser Zeile an. Schon beim ersten Befehl werden wir fündig. Der Befehl MODE 3 wird nicht ausgeführt, weil die 3 nicht angenommen wird.

Die Zeilen 30 bis 50 stellen eine Schleife dar, in der eine einfache Berechnung abläuft. Diese Berechnung hat nur einen kleinen Schönheitsfehler. Es wurde nicht mit dem Befehl DIM ausreichend dimensioniert.

Den folgenden Syntax-Error in Zeile 60 sollten Sie schon selbst finden.

In den Zeilen 70 bis 110 werden Daten mit dem Befehl READ und einer Schleife eingelesen. Die Zeile 70 stellt diese Daten zum Lesen bereit. In diesem Zeilenbereich ist eine ganze Menge nicht richtig!

In der Zeile 80 wird die Schleife zum Lesen von insgesamt 9 Daten geöffnet. Da aber in der Zeile 70 nicht genug Daten zum Lesen bereitgestellt werden, kommt es später beim READ-Befehl zu einem Fehler.

Nachdem die Zahl aus der DATA-Zeile gelesen wurde, wird sie mit Hilfe der Zeile 100 mit PRINT ausgegeben. Komischerweise erscheint anstatt der Zahl immer nur eine Null. Warum dies so ist, sollen Sie selbst feststellen.

In der Zeile 110 wird die geöffnete Schleife wieder geschlossen. Leider stimmt auch hier etwas nicht. Wenn Sie sich ansehen, wie die Schleife geöffnet wurde, werden Sie auch diesen Fehler schnell erkennen.

Die Zeile 120 stellt eine Warteschleife dar, in der auch irgendwie der Wurm drin steckt. Wenn Sie den Fehler in der Warteschleife gefunden haben, kümmern Sie sich anschließend um den Befehl GOTO, denn auch hier wird später ein Fehler erscheinen.

Nachdem Sie nun Ihr Programm entsprechend korigiert haben, können die Zeilen zur Lokalisierung der Fehler wieder gelöscht werden, denn sie haben ihre Pflicht erfüllt.

Soweit zu diesem Beispiel. Wichtig ist nur, daß man sich über die Bedeutung der einzelnen Befehle im Klaren ist.

Ein Trost zum Schluß. Bei besonders hartnäckigen Fehlern in kleineren Programmen schreiben Sie das Programm am besten nochmal. Oft geht dies schneller als alles zum 5. oder 6. Mal zu kontrollieren.

Oft sieht man nämlich den "Wald vor lauter Bäumen nicht" !!

23 Tips und Tricks zum Schneider CPC 464

Für viele von Ihnen, die öfter Programme schreiben, ist ein Drucker wahrscheinlich eine sinnvolle Ergänzung zum System. Auf einen Schlag lassen sich Programm-Listings und andere Ausgaben zu Papier bringen.

Welchen Typ aber soll man sich anschaffen?

Hier kommt es in erster Linie darauf an, was Sie überhaupt mit Ihrem Schneider machen wollen. Wer nur so zum Spaß am Computer seinen Schneider betreibt, ist mit einem preiswerten Matrix-Drucker recht gut bedient. Solche Drucker mit CENTRONICS-Schnittstelle gibt es ab ca. 500 DM aufwärts zu kaufen. Matrixdrucker drucken mit ganz feinen Nadeln, deshalb setzt sich ihr Schriftbild auch aus lauter kleinen Punkten zusammen. Beim Kauf gilt es hier in erster Linie, Schriftbild und Preis miteinander zu vergleichen.

Für diejenigen unter Ihnen, für die das Schriftbild des Matrixdruckers nicht in Frage kommt, gibt es sogenannte Typenraddrucker. Das sind Drucker, die die Schriftqualität von Schreibmaschinen haben. Sie sind in der Regel teurer als Matrixdrucker, lauter und langsamer. Ihr einzigster Vorteil ist daher nur die bessere Schriftqualität.

Mittlerweile gibt es heute viele Schreibmaschinenhersteller, die den Trend zum Computer erkannt haben und ihre elektronischen Schreibmaschinen mit der Standardschnittstelle CENTRONICS ausgerüstet haben. Sie benötigen also nur noch ein Kabel und schon kann es losgehen mit der Ausgabe von beliebigen Daten. Diese Kabel bekommt man fertig am günstigsten im Versandhandel und in den Computerabteilungen der Kaufhäuser. Nur wer mit dem Lötkolben umgehen kann, sollte sich mit der Anfertigung eines Druckerkabels selbst beschäftigen. Wenn man sich die Materialkosten betrachtet, ist das natürlich die preiswerteste Möglichkeit, um an ein solches Kabel zu kommen.

Wer professionell mit seinem Schneider arbeiten möchte, kommt um die Anschaffung eine Diskettenlaufwerks nicht herum. Aber auch hier hat man wieder die Qual der Wahl. Soll es vielleicht ein Originallaufwerk direkt von Schneider sein? Welches Diskettenformat soll man verwenden?

Durch die Verwendung von 3-Zoll-Disketten wird es später wahrscheinlich schwer werden, sogenannte CP/M-Programme zu verarbeiten. Diese Programme liegen z.Zt. fast alle nur im 5.25-Zoll-Format vor und müßten extra umkopiert werden.

Falls Sie sich ein Laufwerk kaufen möchten, sind Sie meiner Meinung nach mit dem Originallaufwerk mit dem Format 3 Zoll der Fa. Schneider am Anfang erst einmal gut bedient. Wie dann die weiteren Entwicklungen aussehen, wird sich zeigen müssen.

Vielleicht ist Ihnen ja auch bekannt, daß Sie ein zweites Laufwerk der Größe 5.25 Zoll als Zusatzlaufwerk anschließen können. Dieses Laufwerk lohnt sich aber erst für Personen, die mit beiden Diskettenformaten arbeiten müssen und sehr viel programmieren.

Soweit zu den Hardware-Tips. Und nun noch ein paar Tricks für die Programmierung des Schneiders.

Einige wichtige CALL-Aufrufe dienen zur besseren Nutzung des Systems und sollen deshalb hier noch aufgeführt werden.

a.	`CALL &BB06`	Warten auf eine Taste (Vergleichbar mit INKEY$)
b.	`CALL &BB9C`	Schaltet die Ausgabe auf INVERS um. Entspricht dem `PRINT CHR$(24)`.
c.	`CALL &BB8D`	Schaltet den Cursor ab.
d.	`CALL &BB8A`	Cursor wieder einschalten.
e.	`CALL &BB6E`	Schaltet das Relais des Motors ein.
f.	`CALL &BB71`	Schaltet dieses Relais wieder ab.

Die letzen beiden CALL-Aufrufe lassen sich beispielsweise zu Steuerzwecken verwenden, wenn man diesen Relaisausgang für die Ansteuerung einer Elektronik verwendet. Dazu braucht man nur eine Leuchtdiode parallel zum Relais zu schalten, die einen Optpkoppler ansteuert. Mit diesem Optokoppler können Sie Signale weiter verarbeiten, ohne die Gefahr, daß am Schneider etwas kaputt geht. Durch den Optokoppler sind beide Stromkreise galvanisch voneinander getrennt. Dies führt zu keinen internen Schwierigkeiten. Aber Vorsicht! Sie verlieren dabei Ihre Garantie!!

24 Weitere Anregungen und Ausbaumöglichkeiten

Nachdem Sie sich nun einen Überblick über die Programmierung verschafft haben, können Sie nun selbst einschätzen, ob Ihnen die Arbeit mit dem Computer Spaß macht oder nicht. Irgendwann kommt dann nämlich der Zeitpunkt, wo die Frage auftritt: Was nun mit dem Computer?

Glauben Sie bitte nicht, daß Ihnen auf Anhieb sofort die tollsten Ideen kommen, was man als nächstes machen kann! Auch ich stand vor diesem Problem nachdem ich vor vielen Jahren in die Computerei einstieg.

Ich habe mich damals auf Anwendungsprogrammierung für den technischen und kaufmännischen Bereich spezialisiert, als ich die Lust an der Programmierung von Videospielen verloren hatte. Vor einigen Jahren gab es für den Bereich der kleineren Computer kaum Programme. Da hatte man es schon leichter, sich mit etwas zu beschäftigen, was es noch nicht auf dem Softwaremarkt gab. Heute bekommen Sie für fast jeden Zweck ein fertiges Programm, daß Sie nur noch erwerben müssen und schon kann die Anwendung losgehen.

Was kann man also noch alles programmieren, wenn man Spaß an der reinen Programmierung hat? Zuerst würde ich versuchen, alle meine Programme benutzerfreundlich zu gestalten. Gerade dies ist ist heute noch ein sehr großer Schwachpunkt, auch bei käuflichen Programmen. Versuchen Sie sich doch einmal daran.

Wer mehr Spaß an Videospielen hat, dem kommen die guten Grafikeigenschaften des Schneiders auch hier entgegen. Versuchen Sie doch mal die Programmierung eines einfachen Spiels mit selbst definierten Spielfiguren!

Eine ganz neue Art von Videospielen sind sogenannte ADVENTURES, daß sind Spiele, bei denen man sich noch einige Gedanken während des Spielverlaufs machen muß. Programmieren Sie doch einfach mal ein kleines ADVENTURE.

Wer sich mehr für die Programmierung von Anwendungen interessiert, für den gibt es folgende Möglichkeiten. Für den gesamten kaufmännischen Bereich gibt es noch viele kleine Einsatzmöglichkeiten, z.B.

Kalkulationen oder Rechnungsschreibprogramme. Denkbar sind auch Programme, die vielleicht die Angebotsscheibung für Handwerker oder andere Selbstständige übernehmen können.

Auch ein Programm zur Textverarbeitung und Datenspeicherung ist eine dankbare Programmieraufgabe, die man in Angriff nehmen könnte, falls man sich nicht eine zulegt.

Für den technischen Bereich sind Statistiken und andere Tabellen von Interesse. Auch hier kann wieder ein Programm zur Datenspeicherung gebraucht werden. Für technische Berichte ist wieder die Textverarbeitung eine sinnvolle Angelegenheit.

Auch im Bereich der Medizin und Ernährung können Sie Ihren Schneider sinnvoll einsetzen. Beispielsweise könnte sich ein Diabetiker Menüvorschläge per Computer machen lassen, in dem unter anderem auch die Angabe der sogenannte BE (Broteinheiten) eine wichtige Rolle spielen.

Wer sich als Schriftsteller versuchen möchte, der kommt um die erwähnte Textverarbeitung nicht herum.

Eine sehr interessantes Gebiet ist das Lernen mit und am Computer geworden! Viele Kinder und Jugendliche haben sowieso schon Interesse am Computer. Warum sollten sie ihn nicht auch praktisch einsetzen? Ein Programm zur Abfrage von Vokabeln oder zum Trainieren der Mathematikaufgaben ist recht schnell erstellt. Haben sie dann noch Spaß an der Programmierung, kann man ja vielleicht mit den Kindern zusammen ein solches Programm entwickeln.

Neben dem Erlernen von Fremdsprachen ist es heute für viele Menschen von beruflichem Vorteil, wenn sie Schreibmaschine schreiben können. Ein Programm zum Erwerb von Schreibmaschinenkenntnissen kann man ohne weiteres selbst programmieren. Sie sehen also, Betätigungsmöglichkeiten gibt es genug!

Wer an der Programmierung nicht so viel Interesse hat, kann auf eine recht stattliche Anzahl von Programmen zurückgreifen. Woher bekommt man nun diese Programme? In jeder guten Buchhandlung gibt es heute eine Fülle von Computer-Fachzeitschriften, in denen die vielen Kleinanzeigen für Sie bestimmt interessant sein werden. Hier können Sie tauschen, verkaufen und Kontakte knüpfen.

Warnen möchte ich Sie vor dem Ankauf sogenannter Raubkopien, die teilweise nur den Bruchteil des Normalpreises kosten. Sie erkennen Raubkopien daran, daß meistens keine Anleitung für das Programm vorliegt und der Datenträger nicht mit dem Original übereinstimmt.

Außerdem schlafen die Softwarehersteller nicht mehr! In jüngster Vergangenheit sind mehrere Softwarefirmen vor Gericht gegangen und haben immer gewonnen. Für den Raubkpierer bedeutet dies die Beschlagnahme von allem, was mit Computern zu tun hat und eine saftige Geldstrafe.

Ein Programm ist heute recht schnell kopiert, denn einen sicheren Kopierschutz gibt es nicht. Wer dann aber meint, er könnte das schnelle Geld durch den Verkauf von kopierten Programmen machen, der fällt über kurz oder lang auf und muß mit den erwähnten Konsequenzen rechnen. Kopieren von urheberreehtlich geschützten Programmen lohnt sich also im Endeffekt nicht.

Neben der Programmierung ist die Steuerung von Maschinen eine weitere Möglichkeit, den Schneider CPC 464 sinnvoll einzusetzen. Durch den weitverbreiteten Prozessor des Schneiders (dem Z 80), ist die Steuerung von elektrischen und elektronischen Geräten keine Utopie mehr. An diese Sache sollten Sie sich aber erst machen, wenn Sie sich im Bereich Elektronik gut auskennen oder die Möglichkeit haben, auf anschlußfertige Hardwareerweiterungen zurückzugreifen. Die Steuerung eines Industrierobotermodells wäre schon eine interessante Angelegenheit, bei der man auch gleich die Arbeitsweise von Handhabungsgeräten (Robotern) kennenlernen kann. Was Sie aber nun wirklich machen, bleibt eigentlich nur Ihrer Phantasie überlassen.

Anhang

Hier sind noch einmal die wichtigsten Tabellen, Übersichten, Symbole, reservierte Namen und technische Daten aufgeführt.

A ASCII-Code incl. Steuerzeichen

ASCII-Code	Zeichen/Bedeutung
0	hat beim Schneider keine Bedeutung
1	druckt die Steuerzeichen als Grafiksymbole (PRINT CHR$(1);CHR$(7))
2	schaltet den Textcursor aus
3	schaltet den Textcursor wieder ein
4	arbeitet wie MODE, z.B. PRINT CHR$(4);CHR$(0) schaltet auf MODE 0 um
5	setzt ein Zeichen auf die Position des Grafikcursors. Beispiel: MOVE 200,200: PRINT CHR$(5);CHR$(233)
6	schaltet den Textbildschirm wieder ein
7	erzeugt einen Piepton. Beispiel: PRINT CHR$(7)
8	setzt den Textcursor ein Zeichen zurück
9	setzt den Textcursor ein Zeichen vor
10	setzt den Textcursor eine Zeile nach unten
11	setzt den Textcursor eine Zeile höher
12	löscht den Bildschirm wie CLS
13	setzt den Cursor an dem Anfang der gerade beschriebenen Zeile
14	entspricht dem Befehl PAPER
15	entspricht dem Befehl PEN
16	löscht das Zeichen, auf dem der Cursor gerade steht und füllt diese gelöschte Stelle mit der Hintergrundfarbe
17	löscht eine Bildschirmzeile wieder bis zum Anfang zurück
18	löscht den Rest einer Bildschirmzeile bis zum Ende und ist abhängig vom jeweiligen Bildschirmmodus
19	löscht vom Anfang der Zeile bis zu diesem Aufruf alle Zeichen und füllt diese mit der Farbe des Hintergrunds auf
20	löscht ab der Cursorposition bis zum Ende der Zeile alles und füllt diese Stellen mit der Hintergrundfarbe auf
21	schaltet des Textbildschirm ab
22	Transparentmodus ein-/ausschalten
23	setzt den Grafikfarbstiftmodus
24	vertauscht die beiden Farben für PEN und PAPER
25	entspricht dem Befehl SYMBOL
26	entspricht dem Befehl WINDOW
27	ohne Bedeutung
28	entspricht dem Befehl INK
29	entspricht dem Befehl BORDER
30	setzt den Cursor in die linke, obere Ecke (HOME). Bildschirminhalt geht dabei nicht verloren
31	entspricht dem Befehl LOCATE

Es folgen als nächstes alle Zeichen mit Ausnahme der Grafikzeichen ab ASCII-Code 127

ASCII-Code	Zeichen/Bedeutung	Tastaturnummer	
32	Leerzeichen (Space-Taste)	47	
33	!	64	
34	"	65	
35	#	57	
36	$	56	
37	%	49	
38	&	48	
39	'	41	
40	(	40	
41	)	33	
42	*	29	
43	+	28	
44	,	39	
45	-	25	
46	.	31	7*
47	/	30	
48	0	32	15*
49	1	64	13*
50	2	65	14*
51	3	57	5*
52	4	56	20*
53	5	49	12*
54	6	48	4*
55	7	41	10*
56	8	40	11*
57	9	33	3*
58	:	29	
59	;	38	* für die Tasten auf dem Ziffernblock
60	<	39	
61	=	25	
62	>	31	
63	?	30	
64	@	26	
65	A	69	
66	B	54	
67	C	62	
68	D	61	
69	E	58	
70	F	53	
71	G	52	
72	H	44	
73	I	35	
74	J	45	
75	K	37	
76	L	36	
77	M	38	
78	N	46	
79	O	34	
80	P	27	

ASCII-Code	Zeichen/Bedeutung	Tastaturnummer
81	Q	67
82	R	50
83	S	60
84	T	51
85	U	42
86	V	55
87	W	59
88	X	63
89	Y	43
90	Z	71
91	[	17
92	\	22
93	]	19
94	^	24
95	_	32
96	`	--
97	a	69
98	b	54
99	c	62
100	d	61
101	e	58
102	f	53
103	g	52
104	h	44
105	i	35
106	j	45
107	k	37
108	l	36
109	m	38
110	n	46
111	o	34
112	p	27
113	q	67
114	r	50
115	s	60
116	t	51
117	u	42
118	v	55
119	w	59
120	x	63
121	y	43
122	z	71
123	{	17
124	|	26
125	}	19
126	~	--

Die Grafikzeichen bekommen Sie auf den Bildschirm mit dem schon bekannten Programm:

```
10    CLS
20    FOR I=127 TO 255
30    PRINT"ASCII-Code: ";I;" Zeichen:  ";CHR$(I)
40    E$=INKEY$: IF E$="" THEN 40
50    NEXT I
```

Für die Definition des deutschen Zeichensatz sind die ASCII-Codenummern 91 bis 96 sowie die Nummern 123 bis 130 gut geeignet.

B Reservierte Variablennamen

Die hier aufgeführten Namen verwendet das System selbst, Sie dürfen daher nicht für eigene Operationen benutzt werden. Denkbar sind aber Variablenbezeichnungen, in denen Teile von reservierten Wörtern vorkommen. Beispielsweise bei dem Variablennamen PRINTER$ oder STOPEN oder BEENDEN. Solche Bezeichnungen sind zulässig und können teilweise diese Befehle verwenden. Es folgt in alphabetischer Reihenfolge die Liste der reservierten Namen:

```
A: ABS, AFTER, AUTO, AND, ASC, ATN
B: BIN$, BORDER
C: CALL, CAT, CHAIN, CHAIN MERGE, CHR$, CINT,
   CLEAR, CLG, CLOSEIN, CLOSEOUT, CLS, CONT, COS,
   CREAL
D: DATA, DEF FN, DEFINT, DEFREAL, DEFSTR, DEG,
   DELETE, DI, DIM, DRAW, DRAWR
E: EDIT, EI, ELSE, END, ENT, ENV, EOF, ERASE,
   ERL, ERR, ERROR, EVERY, EXP
F: FIX, FN, FRE, FOR
G: GOTO, GOSUB
H: HIMEM, HEX$
I: INP, IF, INPUT, INK, INKEY$, INKEY, INSTR, INT
J: JOY
K: KEY, KEY DEF
L: LEFT$, LEN, LET, LINE INPUT, LIST, LOG, LOAD,
   LOCATE, LOG10, LOWER$
M: MAX, MIN, MOVER, MOD, MEMORY, MODE, MERGE,
   MOVE, MID$
N: NEW, NEXT, NOT
O: ON, ON GOSUB, OPENOUT, OPENIN, ON GOTO, OR, ON
   BREAK GOSUB, ON BREAK STOP, ON ERROR GOTO, ON
   AQ GOSUB, OUT, ORIGIN
P: PAPER, PEEK, PEN, PI, POS, POKE, PLOT, PLOTR,
   PRINT, PRINT USING
R: RAD, REM, RESUME, ROUND, RANDOMIZE, REMAIN,
   RETURN, RUN, READ, RENUM, RIGHT$, RELEASE,
   RESTORE, RND
S: SAVE, SGN, SIN, SOUND, SPACE$, SPC, SPEED INK,
   SPEED KEY, SPEED WRITE, SQ, SQR, STEP, STOP,
   STR$, STRING$, SWAP, SYMBOL, SYMBOL AFTER
T: TAB, TEST, TO, TAG, TESTR, TROFF, TAGOFF,
   THEN, TRON, TAN, TIME
```

```
U: USING, UPPER$, UNT
V: VAL, VPOS
W: WAIT, WEND, WHILE, WIDTH, WINDOW, WINDOW SWAP,
   WRITE
X: XPOS, XOR
Y: YPOS
Z: ZONE
```

Alle die hier aufgeführten BASIC-Befehle dürfen nicht als Variablenbezeichnungen benutzt werden.

C Mögliche Farben und augenfreundliche Farbkombinationen

Mögliche Farben mit Farbnummern

Farbnummer	Bezeichnung der Farbe
0	schwarz
1	blau
2	hellblau
3	rot
4	magenta
5	hellviolett
6	hellrot
7	purpur
8	magentahell
9	grün
10	blaugrün
11	himmelblau
12	gelb
13	weiß
14	pastellblau
15	orange
16	rosa
17	pastellmagenta
18	hellgrün
19	seegrün
20	hellblaugrün
21	limonengrün
22	pastellgrün
23	pastellblaugrün
24	hellgelb
25	pastellgelb
26	hellweiß
27	weiß

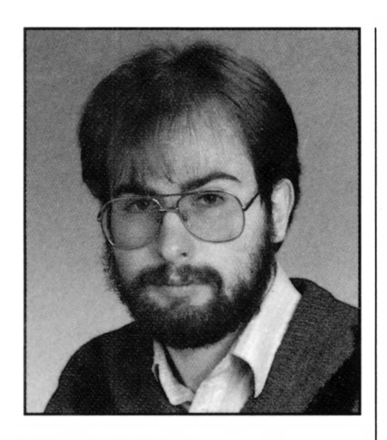
Angenehme Farbkombinationen

Option	Rahmenfarbe	-Nr.	Hintergrundf.	-Nr.	Schriftfarbe	-Nr.
A	schwarz	0	schwarz	0	hellrot	6
B	schwarz	0	schwarz	0	grün	9
C	schwarz	0	schwarz	0	blaugrün	10
D	schwarz	0	schwarz	0	gelb	12
E	schwarz	0	schwarz	0	weiß	13
F	schwarz	0	schwarz	0	orange	15
G	schwarz	0	schwarz	0	limonengrün	21
H	schwarz	0	schwarz	0	pastellgrün	22

D Gebräuchliche Symbole für Programmablaufpläne

Symbol	Erklärungen / Bemerkungen / Beispiele
	Programmstart und Programmende
	Übergangsstelle zu einem anderen Programm Beispiel: Datenschnittstelle
	Allgemeine Operationen, z. B. eine Berechnung
	Logischer Vergleich mit nur zwei Ausgängen Ausgang 1: Ja, Vergleich trifft zu Ausgang 2: Nein, Vergleich trifft nicht zu
	Unterprogramm, z. B. Tastaturabfrage
	Dateneingabe oder Datenausgabe Beispiele: Mit PRINT etwas ausgeben, mit INPUT etwas eingeben
	Datenflußlinien

Stichwortverzeichnis



THOMAS ERPEL
geboren am 8. 1. 1960, arbeitet seit mehr als sieben Jahren mit Rechnersystemen verschiedener Hersteller, so zum Beispiel TANDY, APPLE, COMMODORE, SHARP und SINCLAIR.
1984 machte er sein Hobby zum Beruf und eröffnete eine Computerschule.
Neben den Schulungen werden noch Beratungen durchgeführt und Individual-Software für IBM-PC und kompatible Systeme erstellt.

CPC BASIC-Kurs

Der Leser wird systematisch – bei Nutzung von vielen lehrreichen Beispielen – mit der Programmiersprache BASIC vertraut gemacht. Beginnend mit den Grundlagen wie der Erklärung der Tastatur, einfacher BASIC-Worte und deren Anwendung, Schleifenprogrammierung oder der Datenspeicherung auf Kassette und Diskette werden die wichtigsten BASIC-Standardbefehle ausführlich und in einer für den Computer-Neuling verständlichen Sprache erklärt.
Der zweite Teil des Buches beschäftigt sich mit den Möglichkeiten des Schneider-BASIC. Anhand von unterhaltsamen Übungsbeispielen werden die komfortablen Befehle für die Programmierung von Grafik, Sound und Windows dargestellt. Eventuell auftauchende Anfangsschwierigkeiten lassen sich mit Hilfe der ausführlich kommentierten Beispiele einfach überwinden; die angewandte Syntax zeigt die Praxisbezogenheit des Buches und verleiht ihm dadurch den Charakter eines Nachschlagewerkes. Besonders interessant für den bereits Fortgeschrittenen sind die mehr als 20 fertig programmierten Bausteine, die sofort für eigene Programme verwendet werden können.
Weitere Kapitel behandeln die Fehlersuche in BASIC-Programmen, deren Ursachen und mögliche Beseitigung, strukturiertes Programmieren mit Programmablaufplänen und letztendlich Tips und Tricks, die für jeden Schneider-CPC-Anwender nützlich sein dürften.
Beim Verlag ist eine Kassette mit allen im Buch vorgestellten Programmen unter der Bestell-Nr. MT 846 zu beziehen.

ISB N 3-89090-167-0



Markt&Technik
Verlag Aktiengesellschaft

DM 46,-
sFr. 42,30
öS 358,80

828

Erpel

CPC BASIC-Kurs

M&T


AMSTRAD
CPC
OCR
MÉMOIRE ÉCRITE
MEMORY ENGRAVED
MEMORIA ESCRITA
ACME
https://acpc.me/